Dash Enterprise and Chart Studio Enterprise v4.0.1

Dash On-Premise 4.0.1 Instructions

If you run into any problems, we are here to help!
Configure listeners on ports 443, 80, and 3022 to point to instance ports 443, 80, and 3022, respectively.

1. Before you install
   1.1 Domain name requirements
   1.2 TLS/SSL certificates
      1.2.2 A note on self-signed certificates
   1.3 Docker requirements for running Dash Enterprise
   1.4 External service information
      1.4.1 Email server integration
      1.4.2 LDAP and SAML authentication
      1.4.3 Mapbox integration
   1.5 Types of installation
      1.5.1 Dash Enterprise Single Server: Amazon AMI
      1.5.2 Dash Enterprise Single Server: Other cloud provider or on-premises server
      1.5.3 Dash Enterprise Single Server: Airgapped installation
      1.5.4 Dash Enterprise Single Server: VMware (non-Linux) installation
      1.5.5 Dash Enterprise Kubernetes installation

2. Dash Enterprise Single Server: installation on cloud provider
   2.1 Supported providers
   2.2 Installation on Amazon Web Services
      2.2.1 Associate an Elastic IP
      2.2.2 Administration via SSH
      2.2.3 Configuring a proxy

3. Dash Enterprise Single Server: on-premise installation on own server
   3.1 Server requirements
      3.1.1 Supported Linux distributions
      3.1.2 Additional requirements and recommendations
      3.1.3 Open ports
   3.2 Installation using Plotly script

4. Dash Enterprise Single Server: fully offline installation
   4.1 Dash app limitations in airgapped mode
   4.2 Prepare the environment
      4.2.1 Install Docker
      4.2.2 Additional Docker setup for CentOS 7/RHEL 7 systems
4.2.3 Install the Server Manager
4.2.4 Create Plotly data directory
4.3 Download, rename, and copy installation package
4.4 Install
4.5 Installing Dash Enterprise packages (e.g., Dash-Design-Kit) on an airgapped host

5.1 Hypervisor requirements
5.2 Provision virtual machine
5.3 Configure using the VM’s console

6. Dash Enterprise Kubernetes installation
NVIDIA RAPIDS GPU Support
6.1 Prerequisite knowledge
6.2 Set up your cloud infrastructure
   6.2.1 Amazon Web Services resource prerequisites
   6.2.2 Google Cloud Platform resource prerequisites
   6.2.3 Microsoft Azure resource prerequisites
6.3 Configure additional settings (all providers)

7. Configuration
7.1 Set up your Server Manager
7.2 Run preflight checks
   7.2.1 Preflight check failure: no route to host
7.3 Configure basic settings
   7.3.1 Authentication settings
      7.3.1.1 LDAP authentication
      7.3.1.2 SAML authentication
    7.3.2 Dash Enterprise Kubernetes additional required configuration
       7.3.2.1 Configure external services and enable High Availability for Dash
       7.3.2.2 Update base domain DNS record to use the load balancer IP
7.4 Configure optional settings (if applicable)
7.5 Other options and customizations
   7.5.1 Customize the React Chart Editor (RCE)
7.6 Using Snapshots
   Configuring snapshots
   Taking a snapshot
   Restoring from snapshot
7.7 Running and logging into Dash Enterprise
7.8 Next steps
8. **Upgrade Dash Enterprise**
   8.1 Prepare for upgrade
   8.2 Upgrade
      8.2.1 Installation on cloud provider or installation on own server
      8.2.2 Fully offline (airgapped) installation
      8.2.3 VM Edition
   8.3 Configure the upgraded instance
   8.4 Run the upgraded instance
   8.5 Online migrations
   8.6 Clean up after the upgrade

9. **Client library setup**
   9.1 Python
   9.2 R
   9.3 MATLAB
   9.4 Other libraries/plugins

10. **General troubleshooting and problem solving**
   10.1 LDAP: Testing authentication
   10.2 Testing without adding entries to your DNS server (using /etc/hosts)
   10.3 SSL Error: certificate verify failed when using the Plotly Python library
   10.4 Private key/certificate debugging
   10.5 Running devicemapper in loopback mode
   10.5 Dash app manager boot failure -- Error response from daemon: max depth exceeded
   10.6 Plotly data directory verification failure
   10.7 Missing container-selinux dependency (Error: nothing to do)
   10.8 OCI runtime error
   10.9 Dash Enterprise Kubernetes setup problems
   10.9 Support
      10.9.1 Directly upload support bundle to our portal (recommended)
      10.9.2 Download and e-mail support bundle (for airgapped instances)

11. **Load balancer configuration**
   11.1 Initial considerations
   11.2 Port setup
   11.3 Listener setup
   11.4 Domain (DNS) setup

12. **Reverse proxy setup instructions**
1. **Before you install**

Dash Enterprise is a powerful data visualization platform designed to run on its own server or virtual machine. To support your needs, we offer different methods of installing the software, but the below requirements apply to all methods.

1.1 **Domain name requirements**

Dash Enterprise needs at least one and up to three domain names and corresponding DNS entries, depending on the features you want to enable:

1. **Base Domain Name**: If using the optional Chart Studio Enterprise component, this is the address used to access it (example: `plotly.your-domain.com`)
2. **Streaming Domain Name**: Only required if using the streaming feature (example: `plotly-stream.your-domain.com`)
3. **Dash Domain Name**: Required to use Dash Enterprise; users will visit this address to use Dash Enterprise (example: `dash.your-domain.com`)
   - Individual Dash apps will be available at `https://dash.your-domain.com/appname`

The DNS entries can be either CNAME or A records, but all must be **fully qualified domain names**. If using Dash Enterprise Single Server, the DNS entries for these domains should point to the single server's IP address. If using a load balancer (see **Section 11: Load balancer configuration**) or installing Dash Enterprise Kubernetes, the DNS entries should point to the load balancer IP.

To use Dash Enterprise while waiting for domain names to be assigned or DNS entries to be added, the following work-arounds are available:

1. **Use Plotly-hosted DNS service**: We can add DNS entries for you as a subdomain of our DNS zone (`plotly.host`)
   - First, ensure your server can reach `plotly.host` by testing as follows:
     i. In a web browser on the server, visit `https://yes.plotly.host` (you should see a page with the Plotly logo and the message: *Yes, plotly.host works*); or
     ii. In a terminal, run the following command on the server you plan on using for Dash Enterprise (Again, you should see: *Yes, plotly.host works*):

        ```
curl https://yes.plotly.host
        ```
   - Then, contact our On-Premise Support Team with your server's IP address (either public or private) and the domain name you would like
     i. Please allow up to 24 hours for DNS changes to become effective
2. **Use local hostname:** If not using streaming, Dash Enterprise, or Dash Enterprise Kubernetes, you can enter the server's existing hostname as the **Base Domain Name** in the Server Manager Settings (if it is a fully qualified domain name)
   - To find out the server's hostname, open a console and run `hostname -f`

3. **Use your network’s hosts file:** For testing purposes, you can create the DNS mappings yourself and add them to your server’s `hosts` file (see Section 9.2: Testing without adding entries to your DNS server)

### 1.2 TLS/SSL certificates

All access to Dash Enterprise is over HTTPS web connections secured using TLS/SSL, and Dash Enterprise will redirect all HTTP requests to HTTPS if the appropriate ports are opened. If necessary, Dash Enterprise will also generate its own self-signed certificates; however, using self-signed certificates will require your users to suppress security warnings in their web browsers and other software. To avoid this, we recommend obtaining TLS/SSL certificates issued by a certificate authority (CA).

If you are using a load balancer, it can be used to perform TLS/SSL termination. In this case, adding certificates is not required (see Section 11: Load balancer configuration for more information).

Each domain name you configure requires its own certificate. There are two ways to achieve this:

1. **Multiple certificates:** you can provide a certificate for each of these domain names (up to 3, depending on the features you enable); or
2. **Wildcard certificate:** a single certificate for `*.your-domain.com` can cover all these domains, as long as they are only one level deep:
   - Any domain name of the form `subdomain.your-domain.com` will be covered (for example, `plotly.your-domain.com`, `dash.your-domain.com`)
   - Additional levels of subdomains will not be covered (for example, `dash.yourdepartment.your-domain.com`, `plotly.test.your-domain.com`)

#### 1.2.2 A note on self-signed certificates

Both self-signed and certificates issued by certificate authorities (CAs) allow for secure encryption of the information sent to your Dash Enterprise server. The difference is that, when establishing an HTTPS connection, your web browser will check whether the connection is signed by a recognized CA and will raise a warning if the connection is untrusted (as when secured by a self-signed certificate).

In general, it is not a good idea to become accustomed to suppressing security warnings, since this increases the risk of falling prey to an unauthorized website's impersonation attempt. However, if you are certain the certificate being offered belongs to your Dash Enterprise instance, you can ignore the warning. The safest way of ensuring that the self-signed certificate belongs to your Dash Enterprise instance is to distribute the certificate (`.crt` file) via a trusted connection to all intended users of the site.
1.3 Docker requirements for running Dash Enterprise

Dash Enterprise requires a sufficiently recent version of Docker for smooth operation. Please see Section 12: Docker storage driver requirements and considerations for more information.

1.4 External service information

1.4.1 Email server integration

Dash Enterprise supports email notifications using your own SMTP server. This optional setting allows self-service password resets (if using Plotly-managed authentication) as well as helping keep users informed about collaborators, likes, or comments on plots. To configure outgoing email, see Section 7.4: Optional settings.

1.4.2 LDAP and SAML authentication

Dash Enterprise supports authentication using your company's existing LDAP or SAML authentication system, as well as self-service or administrator-managed account creation modes. You can switch between authentication modes at will. Detailed instructions for authentication configuration are available in Section 7.3: Basic Settings.

When using LDAP, Dash Enterprise supports restricting login to a specified set of groups and sharing files with specified groups.

To use SAML, your SAML identity provider must add Dash Enterprise as a trusted service provider or relying party. Dash Enterprise supports restricting app creator status to members of groups specified on the SAMP IdP, and any individual SAML user can be added as a Dash app collaborator after logging into Dash Enterprise once, which initializes their user account.

1.4.3 Mapbox integration

Mapbox is a third party product that offers high-quality maps and satellite views that can be used with Dash Enterprise. It is not necessary to use Mapbox to access the mapping functionality that ships with Dash Enterprise, and you can also get started without Mapbox and enable it later if desired.

1.5 Types of installation

1.5.1 Dash Enterprise Single Server: Amazon AMI

For all customers using Amazon Web Services, we recommend installing Dash Enterprise using our custom-developed Amazon Machine Image (AMI), since it is the fastest and simplest setup option. To use this, your cloud must be able to access the Internet, either directly or via a proxy. See Section 2: Dash Enterprise Single Server: installation on cloud provider to proceed with this type of installation.
1.5.2 Dash Enterprise Single Server: Other cloud provider or on-premises server

If you need to install Dash Enterprise on an on-premise server or on an instance with a different cloud provider, and that server can access the Internet (either directly or via a proxy), see Section 3: Dash Enterprise Single Server: installation on own server to proceed.

1.5.3 Dash Enterprise Single Server: Airgapped installation

If your server does not have access to the Internet, see Section 4: Dash Enterprise Single Server: fully offline installation. Several limitations apply to Dash apps deployed to a fully offline server; please see the start of that section for details.

1.5.4 Dash Enterprise Single Server: VMware (non-Linux) installation

If you are unable to install directly on a supported Linux server, see Section 5: Dash Enterprise Single Server: VM Edition installation for information on installing the VM Edition, where Dash Enterprise is installed on a VM hypervisor using an OVF package. Because not all versions of Dash Enterprise are available as a VM Edition, and because installation is complex, we recommend this option only as a last resort.

1.5.5 Dash Enterprise Kubernetes installation

Dash Enterprise now includes support for high-availability clusters hosted on Amazon Web Services and Google Cloud Platform. If you are interested in taking advantage of this new feature, please contact the Plotly sales team to discuss further. Otherwise, if you have already acquired a Dash Enterprise Kubernetes license, see Section 6: Dash Enterprise Kubernetes Installation using cloud provider's Kubernetes cluster.

Dash Enterprise Kubernetes is not available in VM Edition.
2. Dash Enterprise Single Server: installation on cloud provider

This section applies to installing Dash Enterprise on a single server on a supported cloud provider. For installing the Kubernetes version of the software, see Section 6: Dash Enterprise Kubernetes installation using cloud provider's Kubernetes cluster.

2.1 Supported providers

Currently we support a simplified installation on Amazon Web Services using an EC2 virtual server and a custom AMI. (Similar support for other cloud providers is in development.) To install on a different cloud provider, simply provision a compute instance that meets our requirements, then follow the instructions in Section 3: Dash Enterprise Single Server: installation on own server.

With the current release, your cloud instance must be able to access the Internet, either directly or via a proxy, to use our automated installation process. If your instance does not have Internet access, skip ahead to Section 4: Dash Enterprise Single Server: fully offline installation.

2.2 Installation on Amazon Web Services

Dash Enterprise can be installed by launching an Amazon Machine Image as follows:

1. Log into your AWS console and visit the EC2 Management Console in the region of your choice; in this example, we have chosen the ca-central-1 region:

2. Click Launch Instance
3. At **Step 1: Choose an Amazon Machine Image (AMI)**, select **Community AMIs** (circle 1) from the menu on the left, then enter **plotly** in the search box (circle 2):

4. Ensure the ID of the AMI (circle 3) matches the ID in the below table for the region you’re using:

<table>
<thead>
<tr>
<th>Region</th>
<th>AMI ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>eu-north-1</td>
<td>ami-05f8e43b2d492e1e4</td>
</tr>
<tr>
<td>ap-south-1</td>
<td>ami-0296883d10b97c8f9</td>
</tr>
<tr>
<td>eu-west-3</td>
<td>ami-06ab12465e5df8aa6</td>
</tr>
<tr>
<td>eu-west-2</td>
<td>ami-058d64fc0cb35432f</td>
</tr>
<tr>
<td>eu-west-1</td>
<td>ami-06df805a04137cffe</td>
</tr>
<tr>
<td>ap-northeast-2</td>
<td>ami-0d74298f951ec258d</td>
</tr>
<tr>
<td>ap-northeast-1</td>
<td>ami-0ed442b942a9fdd9c</td>
</tr>
<tr>
<td>sa-east-1</td>
<td>ami-09d2980f4eb3e337f</td>
</tr>
<tr>
<td>ca-central-1</td>
<td>ami-05ade474e8e05958f</td>
</tr>
<tr>
<td>ap-southeast-1</td>
<td>ami-0589e1c357901e854</td>
</tr>
<tr>
<td>ap-southeast-2</td>
<td>ami-0b55dd6085805c474</td>
</tr>
<tr>
<td>eu-central-1</td>
<td>ami-0f75f1ce46d8dd064</td>
</tr>
<tr>
<td>us-east-1</td>
<td>ami-0809c3b7147f2113f</td>
</tr>
<tr>
<td>us-east-2</td>
<td>ami-07fbf99f258b9e87e</td>
</tr>
<tr>
<td>us-west-1</td>
<td>ami-0442e6fb50a1ace90</td>
</tr>
<tr>
<td>us-west-2</td>
<td>ami-063ad81c361764a57</td>
</tr>
</tbody>
</table>
5. After verifying the AMI ID, click **Select** to choose the AMI.

6. In **Step 2: Choose an Instance Type**, select an appropriate instance type and click **Next: Configure Instance Details**
   - Dash Enterprise requires at least 4 vCPUs and 16 GB RAM, so **t2.xlarge** is a good option; for additional performance, you may choose a larger instance type.

7. On **Step 3: Configure Instance Details**, set **Auto-assign public IP** to **Disable**, then click **Next: Add Storage**.

8. On **Step 4: Add Storage**, set the **Size (GiB)** of the root volume, then click **Next: Add Tags**
   - We recommend **2000** GiB, based on a team of 20 users updating charts daily.
   - This number may be increased or decreased depending on the size of the active team, but should never be less than **200** GiB.
   - We currently do not support splitting your server's directory structure across multiple block devices; please add only one storage volume of the required size.

9. On **Step 5: Add Tags**, you may optionally add a **Name** tag of **plotly-on-premise** and any other tags you require for your organization purposes.
When done, click **Next: Configure Security Group**

10. On **Step 6: Configure Security Group**, select **Create a new security group**, give it a descriptive name such as **plotly-on-premise**, and configure entries as below:

   - The **Custom TCP 3022** rule is only needed for Dash Enterprise; all others are required
   - To restrict access to Dash Enterprise to a single IP address (or range of IPs) replace the default **Source** values (0.0.0.0/0, ::/0) with the address or range
   - Once you have configured the security group, click **Review and Launch**

11. On **Step 7: Review Instance Launch**, carefully review the information and especially confirm that the security groups match the ones shown below:

   - If any information is incorrect, use the **Edit** links to go back and correct it.

12. After verifying, click **Launch and** choose a key pair you have access to, or create a new key pair and download it
This step is extremely important! Without this key pair, you will not be able to SSH into your server and the key pair is very difficult to reset.

13. On the Launch Status screen, click View Instances at the bottom right
14. Associate an Elastic IP with your instance (see Section 2.2.1: Associating an Elastic IP)

2.2.1 Associate an Elastic IP

To ensure your instance's IP address does not change when the instance is rebooted or undergoes other maintenance, associate an Elastic IP address with the instance as follows:

1. In your AWS console, select Elastic IPs from the menu on the left

2. Select Allocate new address, leave the Scope set to VPC, then choose Allocate
3. On the New address request succeeded screen, note the new address (in this case 52.60.140.183) and then click the address

4. In the Elastic IPs screen, ensure your new IP address is selected, then click Actions > Associate address
5. Click **Select an instance** and choose your new instance from the list.

6. Click the **Associate** button at the bottom right, then **Close**

Your instance is now ready for you to upload your license and configure Dash Enterprise. If your instance uses a proxy to access the Internet, see [Section 2.2.3: Configuring a proxy](#). Otherwise, skip to [Section 7: Configuration](#) to continue.

### 2.2.2 Administration via SSH

To administer your instance via SSH, you may log in using the plotly username and the SSH key you selected or uploaded when launching your new instance.

### 2.2.3 Configuring a proxy

If your system requires a proxy to access the Internet, configure as follows:

1. **SSH** into the running instance
2. Run the Plotly installation script:

   ```bash
curl -sSL https://get.plot.ly | sudo bash
   ```
3. When asked if your machine requires a proxy to access the internet, enter y and provide your proxy details when prompted.
4. Once this script is complete, proceed with the instructions in Section 7: Configuration.

If your proxy credentials are rotated periodically, Dash Enterprise can still use your proxy, but please contact Plotly support for additional instructions.
3. Dash Enterprise Single Server: on-premise installation on own server

This section applies to installing Dash Enterprise on a single server hosted on your own network. For installing the Kubernetes version of the software, see Section 6: Dash Enterprise Kubernetes Installation using cloud provider’s Kubernetes cluster.

3.1 Server requirements

To install Dash Enterprise using this method, you need root access to a server that meets the below requirements:

- 64-bit Linux server
- 8CPU (or vCPU) cores
- 16 GB RAM and 32 GB swap space
- 210 GB disk space for the Plotly data directory (2 TB recommended)
- 120 GB of disk space minimum for the Docker data directory (defaults to /var/lib/docker)
- 20 GB of total disk space is a required minimum for the Replicated data directory (/var/lib/replicated)

If using AWS, the server size closest to this is m3.xlarge, but please consider an AMI-based installation in this case.

3.1.1 Supported Linux distributions

We support the server editions of the following distributions or operating systems:

- Debian 7.7+
- Ubuntu 14.04.5 / 16.04 / 18.04 / 20.04
- Red Hat Enterprise Linux 7+
- CentOS 7+
- Oracle Linux 7+

Plotly has the most experience supporting Ubuntu 18.04 and RHEL/CentOS 7, but any of the above are acceptable. If multiple editions exist, choose the server edition and not the desktop edition.

3.1.2 Additional requirements and recommendations

To avoid any issues that may arise as a result of running multiple applications on the same server, we strongly recommend using a dedicated server or VM to run Dash Enterprise.
We do not recommend enabling SELinux on the server.

You must have root access to the server you're using. These instructions assume that you are using the **sudo** utility to gain root access.

To download the required files for installation, the server requires **Internet access** (proxy servers are supported). If Internet access is not an option, we support a fully offline installation; see [Section 4: Fully Offline Installation](#) for instructions.

Ensure the Docker data directory (**/var/lib/docker**) has enough space allocated to it (we suggest at least **100 GB**).

**Certain Docker storage drivers are not supported** as they are either deprecated or not intended to be used in production environments. This includes the **zfs**, **btrfs**, and **devicemapper** (in loopback mode) storage drivers. If you are uncertain whether your Docker storage driver is fully supported, please contact On-Premise Support.

### 3.1.3 Open ports

The following open ports are required:

- **Port 443 (HTTPS)**: Required for creating, viewing, or administering Dash Apps, or the optional Chart Studio component.
- **Port 8800 (Server Manager UI, via HTTPS)**: Required for administrators to install, upgrade, and configure Dash Enterprise. You may restrict access to this port to administrators only.

The following ports are optional but recommended:

- **Port 80 (HTTP)**: All Dash Enterprise requests are made over HTTPS, so this port is not strictly required for full functionality. However, opening this port will allow Dash Enterprise to automatically redirect HTTP requests to use HTTPS; without this port, HTTP requests will simply fail.
- **Port 3022 (Dash app deployment via SSH)**: To support app deployment over SSH, Dash Deployment Server requires a port for SSH connections (default port 3022, configurable in Server Manager Settings). If you do not open a port for Dash app deployment via SSH, you will still be able to deploy Dash apps using HTTPS.
  - We do not recommend changing this to port 22, since you would need to disable SSH on the server.

### 3.2 Installation using Plotly script

1. **SSH into your Linux server**
2. **Create a data directory to hold all of the Dash Enterprise data**
   - We recommend **/plotly** for ease of reference, but you may choose any name that suits your needs
$ sudo mkdir /plotly

3. Run the installation script:

   $ curl -sSL https://get.plot.ly | sudo bash

   ○ If your server uses a proxy to access the Internet, see Section 2.2.3: Configuring a proxy
   ○ If prompted for the server’s service IP address, press Enter to accept the default

Your server is now ready for you to upload your license and configure Dash Enterprise; see Section 7: Configuration to continue.
4. Dash Enterprise Single Server: fully offline installation

This section applies to installing Dash Enterprise on a single server in a fully offline environment. For installing the Kubernetes version of the software, see Section 6: Dash Enterprise Kubernetes installation using cloud provider's Kubernetes cluster.

Dash Enterprise supports fully offline installations (a.k.a. **airgapped** installations), with some limitations in deploying Dash apps (described in Section 4.1: Dash app limitations in airgapped mode).

Apart from Internet access, the server requirements for a fully offline installation are the same as for a normal installation on your own hardware (see Section 3.1: Server requirements), as well as:

1. A supported version of Docker and the Server Manager installed on your server (described in 4.2.3 Install the Server Manager)
2. A link to the Dash Enterprise airgapped installation package, provided to you by the Plotly sales team

4.1 Dash app limitations in airgapped mode

These are currently known limitations with how Dash apps work offline and may be addressed in a future Dash Deployment Server release:

- All Python packages, including Dash libraries and their dependencies, must be available via a private PyPI repository specified in the app's `requirements.txt`
- Only Python versions 2.7.9 - 2.7.14 and 3.6.0 - 3.6.5 are supported
  - If another version is listed in your `runtime.txt`, the app build will hang or fail
- Only Conda runtimes Miniconda2-4.5.12 and Miniconda3-4.5.12 are supported
  - If another version is listed in your `conda-runtime.txt`, the app build will hang or fail
- All Python dependencies must be installed via `requirements.txt` or via Conda
  - Pipfile and `nltk.txt` are not supported.
- `hg+` and `git+` requirements are not supported

4.2 Prepare the environment

4.2.1 Install Docker

We recommend you use the most recent version of Docker supported by your operating system. These operating systems and corresponding Docker versions are known to work together:
1. Downloaded the correct Docker package for your needs, plus any dependencies, and transfer those files to your server
   a. If the server has access to a local yum or apt repository, you can use yum or apt to install dependencies from that repository instead of manually installing them

2. Install Docker using one of the following commands:
   a. CentOS/RHEL/Fedora: sudo rpm -ivh <package_name>.rpm
   b. Ubuntu / Debian: sudo dpkg --install <package_name>.deb

4.2.2 Additional Docker setup for CentOS 7/RHEL 7 systems

After installing Docker, increase the default devicemapper base device size as follows:

1. Add the following line to /etc/default/docker:

   ```
   DOCKER_OPTS=--storage-opt dm.basesize=20G
   ```

   ● If an uncommented `DOCKER_OPTS` line already exists, append the following string to it instead of adding a new line:

   ```
   --storage-opt dm.basesize=20G
   ```

2. Restart Docker:
   a. Debian/Ubuntu: sudo service docker restart
   b. CentOS 7/RHEL 7/Fedora: sudo systemctl restart docker
4.2.3 Install the Server Manager

The Server Manager is a web-based console where you can manage your Dash Enterprise (and, if installed, Chart Studio Enterprise) instance's settings, as well as perform upgrades and start/stop the application. This component is normally installed automatically when using an online installation process, but when working offline, must be installed separately as follows:

1. On an Internet-capable machine, download the latest Server Manager release from https://s3.amazonaws.com/replicated-airgap-work/replicated.tar.gz and transfer it to your destination server.
2. In a terminal, run the following commands:

```bash
tar xzvf replicated.tar.gz
cat ./install.sh | sudo bash -s airgap
```

Once installed, the Server Manager is available on port 8800 of your server's IP or base domain (e.g. https://12.34.56.789:8800 or https://plotly.your-company.com:8800).

4.2.4 Create Plotly data directory

This directory will hold all of the Dash Enterprise data, such as configuration, apps and charts, databases, logs, and other files. You can name this directory anything, but we recommend /plotly for consistency with other installation types and the rest of this documentation.

```bash
sudo mkdir /plotly
```

4.3 Download, rename, and copy installation package

1. Use the download link and password provided to you by Plotly to download an airgapped installation package
   a. Set the --trust-server-names flag for wget
   o Alternatively, rename the file to something ending with .airgap
2. Copy the package to your server and note the path you copied it to

4.4 Install

1. Browse to the Server Manager on port 8800 of your base domain
   ● You will see an untrusted connection warning; this is normal
2. Accept the initial self-signed certificate and enter the Hostname of your server
   ● If you have a CA-issued TLS/SSL certificate and the corresponding key, select them here and choose Upload & Continue
   ● Otherwise, choose Use Self-Signed Cert and accept the new certificate generated
3. Click Choose License and select the license (.rli) file provided to you by the Plotly sales team
4. Choose Airgapped as the installation type and click Continue
5. Enter the full path to the airgap package file you copied in Section 4.3 and click Continue

Your server is now ready to be configured. Skip ahead to Section 7: Configuration to continue.

4.5 Installing Dash Enterprise packages (e.g., Dash-Design-Kit) on an airgapped host

The images used to build Dash applications on the DE server are configured to install Plotly-hosted Dash dependencies (e.g., Dash-Design-Kit) via an internal repository found here: https://PLOTLY_DASH_DOMAIN_BASE/Docs/packages/.

The PLOTLY_DASH_DOMAIN_BASE referenced in this URL is the DNS name configured for the Dash Enterprise server. Additional consideration is often needed on airgapped environments to reach the packages hosted at this endpoint. If PLOTLY_DASH_DOMAIN_BASE is configured to be the Public IP of the host and if this Public IP is unreachable from the host itself, we propose the following two workarounds:

1. Modify the Dash Domain DNS entry to be the private IP of the host instead. A proxy server will need to be used to reach your host from a browser once this change is made.

2. Alternatively, any needed packages found at https://PLOTLY_DASH_DOMAIN_BASE/Docs/packages/ can be downloaded and transferred to an internal Python Package Repository that can be reached from your instance.


This section applies to installing Dash Enterprise Single Server in a local VMware environment. VM Edition is intended for use only by customers who cannot use Docker containers directly. Support for this edition is limited and the latest version of Dash Enterprise is not always available as a VM. If possible, we strongly recommend using the regular Dash Enterprise release.

The below walkthrough assumes that you are using VMware ESXi managed by the VMware vSphere client. If that is not the case, please consult the documentation for your hypervisor.
5.1 Hypervisor requirements

Dash Enterprise requires a hypervisor (also known as virtualization player or virtualization host) that supports OVA/OVF files. This release was tested on VMware vSphere 6 (ESXi) Hypervisor 6.0.0 (managed by VMware vSphere Client), but any other hypervisor that supports OVA/OVF should also work.

The virtual machine to be provisioned requires the following system resources:
- 4 CPU cores
- 16 GB RAM
- 2 TB free disk space (based on a 20-user team updating charts daily
  - This number may be increased or decreased depending on the size of the active team, but should never be lower than 200 GB

5.2 Provision virtual machine

1. Create a new virtual machine using the plotly-on-premise-3.1.0.ova file provided to you by Plotly:
   a. Go to File > Deploy OVF Template
   b. Select plotly-on-premise-3.1.0.ova and click Next
   c. On the Details screen, click Next
      i. The description is incorrect; this is known and is to be fixed in a later release
   d. Name the machine (we suggest Dash Enterprise 3.1) and click Next
   e. On the Summary screen, leave Power on after deployment disabled and click Finish
      i. The virtual machine will now be deployed
      ii. This may take some time if you have a slow network connection, since the OVA file is being uploaded to your hypervisor

2. Configure the virtual machine
   a. Click the new virtual machine in the Inventory and select Edit virtual machine settings
   b. Increase memory to 16 GB (or more if desired)
   c. Increase CPUs to the maximum available
   d. Click Add > Hard Disk > Create a new virtual disk and add a 2 TB disk
   e. Do not change any advanced options

3. Power on the virtual machine and continue configuration using the console (see next section)
   a. The virtual machine may take up to two minutes to boot

5.3 Configure using the VM’s console

To visit the console of your new VM, right-click on the Dash Enterprise entry in your virtual machine inventory and choose Open Console. If the virtual machine has finished booting, the Dash Enterprise banner will be displayed in the console. You will then be taken to the First Boot menu to configure the system as follows.
1. **Set shell password:** this is for the plotly user account, which has root access via `sudo` for system administration
   - The default password is `plotly`
2. **Set root password:** this password will be for the root user and is recommended for emergency repairs to your system
   - If you do not set this password, the root user will be unable to log in
3. **Install Replicated:** this begins the installer for the Server Manager
   - If you have more than one network interface, select the one that will be used to access the server from your network
   - When asked if this machine uses a proxy to access the Internet, enter `y` if this is the case or if it uses a proxy to access your internal networks; otherwise, enter `n`
4. **Prepare storage:** this attempts to automatically locate and prepare data storage for Dash Enterprise, as follows:
   - If there is an existing formatted partition labelled `plotly`, it will be added to `fstab`, mounted, and used for data storage (this supports upgrades)
   - If not, any unpartitioned drives will be partitioned, then the largest unformatted partition will be formatted, labelled as `plotly`, added to `fstab`, and mounted to be used for data storage
     i. If you added a second hard disk or block device meeting these criteria, it will automatically be prepared for use as data storage
   - If you are unable to use the Prepare storage option, see the next step
5. **Perform shell-based configuration if needed:** if you are not able to use the automated installer, you can use the Run a shell utility for configuration tasks
   - To prepare storage manually:
     i. Create a partition and format it
     ii. Mount the partition on `/plotly`
     iii. To support upgrades, add the partition to `/etc/fstab` and label it `plotly`
6. **Continue to web-based configuration:** choose Exit installer to continue the rest of the installation in a Web browser; see **Section 7: Configuration** to continue.
6. Dash Enterprise Kubernetes installation

Dash Enterprise now includes support for high availability clusters hosted on Amazon Web Services and Google Cloud Platform. If you are interested in taking advantage of this new feature, please contact the Plotly sales team to discuss further. Otherwise, if you already have Dash Enterprise Kubernetes enabled on your license, see below for setup instructions.

Please note that support for Dash Enterprise Kubernetes does not include Chart Studio at this time.

**NVIDIA RAPIDS GPU Support**

As of v4.0.0, Dash Enterprise Kubernetes (DEK) comes with support for assigning GPU resources to a scaled app process for **RAPIDS CUDA 10.1** workloads. This can be performed via the `resources` app page within the Dash App Manager, and allows users to scale from 0 to 4 GPUs on an app process.

In order to utilize GPU scaling, your kubernetes cluster must come with the appropriate number of GPUs. Additional instructions have been included in the installation sections below for the supported providers. These additional instructions will be separated out with a **NVIDIA RAPIDS GPU** heading for the AWS and GCP providers.

### 6.1 Prerequisite knowledge

The below walkthrough can be completed entirely using your cloud provider's graphical interface with no knowledge of Kubernetes. However, managing and troubleshooting your cluster can be faster and simpler if you are familiar with common management tasks for Kubernetes and your selected cloud provider. For tutorials on common Kubernetes tasks, please consult the official documentation. The official `kubectl` cheat sheet is also useful as a quick reference for command-line syntax for frequent tasks.

### 6.2 Set up your cloud infrastructure

The below cloud resources must be in place before enabling High Availability mode in the Dash Enterprise Settings. Once you enable High Availability mode, Dash Enterprise will automatically configure the remaining required resources. See below for the prerequisite resources for the cloud provider you are using.
Note: When creating new cloud provider resources, you must assign them to a VPC at the time of creation, as you cannot change a VPC assignment after the resource is created. Because this option is very easy to overlook, please carefully review your resource summaries before confirming their creation.

6.2.1 Amazon Web Services resource prerequisites

1. Create a new VPC (guide) with the following resources:
   a. At least two /20 subnets (guide) covering at least two availability zones, configured to auto-assign public IP addresses (guide)
   b. An Internet Gateway attached (guide)
      ■ You also need to add a route to the VPC’s Route Table that allows all inbound traffic to this Internet Gateway (guide)
   c. DNS resolution and DNS hostnames enabled (guide)
2. Create a new security group (guide) with the following Inbound rules, all with source Anywhere
   a. HTTP
   b. HTTPS
   c. SSH
   d. PostgreSQL
   e. Custom TCP rule: Port 6379
   f. Custom TCP rule: Port 3022
   g. Custom TCP rule: Port 8800
3. Create one IAM Access Management Policy (guide) with the below sets of permissions:
   a. Service: Elastic Container Registry; Actions: All Elastic Container Registry; Resources: Any for repository
   b. Service: EKS; Actions: All EKS actions; Resources: Any for cluster
   c. Service: Systems Manager; Actions: all GetParameter; Resources: Any
4. Create three IAM Service Roles (guide):
   a. One “registry manager” role with the EC2 use case and the new policy from Step 3 attached
   b. One “cluster manager” role created with the EKS - Cluster use case and AmazonEKSClusterPolicy
      ■ After this role is created, click on its name, then Attach Policy and select AmazonEKSServicePolicy
   c. One “worker node” role with the EC2 use case and these policies:
      ■ AmazonEKSWorkerNodePolicy
      ■ AmazonEC2ContainerRegistryReadOnly
      ■ AmazonEKS_CNI_POLICY
5. Create an EC2 instance provisioned with the Plotly AMI according to Section 2.2: Installation on Amazon Web Services, with these Instance Details changed from the default:
   a. Set the Network to the VPC from Step 1 and select a Subnet from that VPC
   b. Set Auto-assign public IP to Disable
   c. Set the IAM role to the “registry manager” EC2 service account from Step 4a
   d. Select an existing security group and assign the group you created in Step 2
Depending on your network, you may need to assign additional security groups; please consult your cloud infrastructure administrator if unsure.

6. SSH into your EC2 instance and:
   a. Install Python 3
   b. Install `aws-cli` v2 (guide) and configure it with at least your AWS Access Key ID and AWS Secret Access Key (guide)
   c. Install `kubectl` (guide)

7. In the Amazon RDS dashboard, create a new DB subnet group in your VPC (guide)

8. Create a single RDS Postgres (version 11) instance with Standard Create (guide)
   a. During configuration:
      ■ Choose the Production template
      ■ Leave the Master username as `postgres` and record the password for use in the next step
      ■ Under Availability & Durability, ensure Create a standby instance is selected
      ■ Under Connectivity, select the VPC you created in Step 1 and its associated VPC security group
      ■ Expand Additional Configuration and set the Initial database name to `streambed`
   b. After the RDS instance is ready, SSH into the EC2 instance you created and connect to the database (guide), then:
      ■ Create a second database named `dash_deployment_server`
      ■ Assign all privileges to the `postgres` user for the `streambed` and `dash_deployment_server` databases

9. In the ElastiCache dashboard, create a new DB subnet group
   a. Use your VPC ID from Step 1
   b. In the Availability Zone or outpost drop-down, select each availability zone you created in Step 1a and Add their Subnet IDs, then click Create

10. Create an ElastiCache cluster using the Redis cluster engine, with cluster mode disabled (guide)
    a. In the Advanced Redis settings, ensure Multi-AZ with Auto-Failover is selected
    b. Select the Subnet group you just created

11. Create an Elastic Kubernetes Service cluster (guide) with the following settings changed from default:
    a. Kubernetes version: 1.15
    b. Cluster Service Role: the “cluster manager” IAM Role created in Step 4b
    c. VPC: the VPC you created in Step 1
    d. Security groups: all groups suggested by the wizard
    e. Cluster endpoint access: private

12. Click on the cluster name, then click the Cluster security group (under the Networking tab) and add an Inbound rule allowing traffic on port 443 from your EC2 instance’s subnet

13. From your EC2 instance’s command line, add your new cluster’s configuration to your kubeconfig (guide)
14. NVIDIA RAPIDS GPU: if you are looking to create a GPU node pool, a modification is needed to step 15. For simplicity, this modification is detailed at the bottom of this subsection after step 17. If this situation applies to you, please read the details of the needed modification before proceeding. If this situation does not apply to you, please continue with step 15.

15. Create an EKS node group (guide) set up as follows:
   a. Node IAM Role Name: the “worker node” IAM Role created in Step 4c
   b. Allow remote access to nodes from selected security groups and choose the group you created in Step 2
   c. Instance type: t3.xlarge
   d. Disk size: 200 GiB
   e. At least 4 nodes

16. From your EC2 instance’s command line, add the Roles you created in Step 4 to the cluster configmap (guide) with the below example
   a. Replace WORKER_NODE_ROLE_ARN with the ARN of the role you created in Step 4c
   b. Replace REGISTRY_MANAGER_ROLE_ARN with the ARN of the role you created in Step 4a
   c. You can retrieve each role’s ARN by visiting its Summary page in the IAM Management Console

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - rolearn: WORKER_NODE_ROLE_ARN
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: REGISTRY_MANAGER_ROLE_ARN
      username: kubectl-access-user
      groups:
        - system:masters
```

17. Create an Elastic Container Registry repository (guide) to contain the images for your Dash apps

∗ NVIDIA RAPIDS GPU: To create a GPU node pool, step 15 must be followed but with this modification: when selecting the AMI, please choose the Amazon Linux 2 GPU Enabled (AL2_x86_64_GPU) AMI. Additionally, the following is a current list of instance families on AWS that support GPUs, and an instance type in one of these families must be chosen for the Managed Node Group:

   a. G2
Once the GPU node pool has been created, you will additionally need to install the NVIDIA GPU device plugin for Kubernetes. This may be installed by following the EKS device plugin documentation: here. Please note that this process may take up to 10 minutes in order to take effect.

With Kubernetes version 1.16 or newer, you may get an error of the form

cannot recognize "https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/1.0.0-beta/nvidia-device-plugin.yml": no matches for kind "DaemonSet" in version "extensions/v1beta1"

when running the kubectl apply command for the NVIDIA device plugin for Kubernetes. If this happens, run the following command to install the plugin:

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/master/deployments/static/nvidia-device-plugin.yml
```

Once step 15 is complete with the above modification, please continue with step 16.

### 6.2.2 Google Cloud Platform resource prerequisites

1. Create a **GCP VPC** that is configured to allow traffic between your cluster nodes as follows:
   a. Inbound on the following ports:
      - 80
      - 443
      - 8800 (Server Manager access)
      - 8008 (internal health checks)
      - 5432 (Postgres)
      - 6379 (Redis)
   b. Dash Enterprise will automatically configure additional firewall rules for the cluster when restarted in High Availability mode
2. Create an **IAM service account** with the following Roles:
   a. Kubernetes Engine Admin
   b. Kubernetes Engine Developer
c. Storage Admin

3. Create a Google Compute Engine instance (guide) with Dash Enterprise installed on it according to our instructions in [Section 3: Dash Enterprise Single Server: On-Premise Installation on Own Server](#).
   
   a. Ensure that you set these values at the time of creation:
      i. The VPC you intend to use from Step 1
      ii. The IAM service account from Step 2

4. Create a single Cloud SQL (guide) instance using Postgres 11, with a Private IP associated with your VPC
   
   a. When creating your instance, under Configuration Options > Backups, recovery, and high availability, set Availability to High Availability
      
   b. Connect to the Postgres instance and create two databases:
      i. streambed
      ii. dash_deployment_server
      
   c. Grant all privileges to the postgres user for both databases

5. Create a Cloud Memorystore Standard Redis instance (guide)

6. NVIDIA RAPIDS GPU: if you are looking to create a GPU node pool, a modification is needed to step 7. For simplicity, this modification is detailed at the bottom of this subsection after step 8. If this situation applies to you, please read the details of the needed modification before proceeding. If this situation does not apply to you, please continue with step 7.

7. Create a Google Kubernetes Engine cluster (guide)
   
   a. Master version: choose Static version 1.15
   
   b. Create one node pool with at least 4 nodes using the n1-standard-4 machine type
   
   c. In Node Pool > More options, select the IAM account you created in Step 2 for the Service account
   
   d. Network: select your VPC from Step 1

8. Create a Google Container Registry repository (guide) for your Dash Enterprise app images

   ✴ NVIDIA RAPIDS GPU: On GKE, GPUs cannot be added to an existing Kubernetes node pool, though new node pools may be associated with a Kubernetes cluster that have GPUs. Also note that there is a GPU quota that must be requested in order to assign GPU resources to a cluster. See the GKE GPU Quota documentation for more details here.

To create a GPU node pool, the following gcloud command can be invoked:

gcloud container node-pools create pool-name
   --accelerator type=gpu-type,count=amount
   --zone compute-zone
   --cluster cluster-name
   --num-nodes 3
   --min-nodes 0
   --max-nodes 5
   --enable-autoscaling
Replace the following and adjust the node size as appropriate:

- **pool-name**: the name you choose for the node pool.
- **gpu-type**: the GPU type. At this time of writing, this list includes: nvidia-tesla-k80, nvidia-tesla-p100, nvidia-tesla-p4, nvidia-tesla-v100, or nvidia-tesla-t4.
- **amount**: the number of GPUs to attach to nodes in the node pool.
- **compute-zone**: the compute zone in which to create the node pool, such as us-central1-c. The cluster must already run in the zone specified.
- **cluster-name**: the name of the cluster in which to create the node pool.

See the GKE documentation on creating GPU node pools for other examples [here](#).

Once the GPU node pool has been created, you will additionally need to install the NVIDIA GPU device drivers. DEK 4.0 supports CUDA 10.1 which requires a minimum driver version of 418.39. The NVIDIA GPU device drivers may be installed by following the GKE driver documentation [here](#). Please note that this process may take up to 10 minutes in order to take effect.

Installing the drivers will automatically taint GPU node pools to disable scheduling of non-GPU workloads only if there is an existing non-GPU node pool. This taint will not apply retroactively to existing GPU nodes if the cluster is created with GPU nodes only and non-GPU node pools are added afterwards.

Once step 6 is complete with the above modification, please continue with step 8.

### 6.2.3 Microsoft Azure resource prerequisites

1. Create a **Resource Group** ([guide](#)) to hold all the resources you will create in steps 2 through 5
2. Create an **Azure Linux virtual machine** ([guide](#)) with Dash Enterprise installed on it according to our instructions in [Section 3: Dash Enterprise Single Server: On-Premise Installation on Own Server](#)
   a. In the **Networking** section, set **NIC network security group** to **advanced** and add inbound ports as described in [Section 3.1.3 Open ports](#)
   b. In the **Management** section, enable **System assigned managed identity**
3. Create an **Azure Database for PostgreSQL** ([guide](#)) instance using Postgres 11 and configure as follows:
   a. In **Connection Security**:
      i. Set **Allow access to Azure services** to **Yes** and add the IP address of the VM you created in Step 2 to the field
      ii. Set **Enforce SSL connection** to **Disabled**
      iii. Under **Allow access to Azure services**, add the IP address of your workstation, then connect to the Postgres instance and create the following two databases:
4. Create an Azure Cache for Redis (guide) and unblock port 6379
5. Create an Azure Kubernetes Service cluster (guide) and configure it as follows:
   a. Kubernetes version: 1.15
   b. Recommended node size: D4_v3
      i. Set this to N12s_v2 if the cluster will be used to deploy RAPIDS apps
   c. Node count: >= 4
   d. Authentication method: System-assigned managed identity
   e. Network configuration: advanced
      i. Use the same virtual network as the resource group
   f. Select Calico network services
   g. In the Integrations section, create a new container registry
6. Assign the virtual machine from Step 2 the Contributor IAM role to the cluster as well as the container registry (guide):
   a. Go to the cluster’s page, select Add a role assignment from IAM roles and select the created VM as a Contributor
   b. Go to the container registry’s page and add the role assignment following similar steps

As part of installation, Azure will create a new resource group with the following naming structure:

\[ MC_{resource\ group\ from\ step\ 1}_{cluster\ name}_{region} \]

To prevent configuration problems, please avoid modifying this group.

### 6.3 Configure additional settings (all providers)

1. In your DNS host, create a A record pointing your Base Domain Name (e.g. plotly.your-domain.com) to your compute instance from Step 3a
   - GCP: create an A record using the instance’s public IP address
   - AWS: create a CNAME record pointing to the instance’s Public DNS name
   - Note that for both providers, you will change this record later while configuring Dash Enterprise settings to enable high-availability mode
2. In your DNS host, create a CNAME record pointing your Dash Domain Name to your Base Domain Name
3. Proceed with additional configuration as outlined in Section 7: Configuration
7. Configuration

Before use, Dash Enterprise must be configured via the Server Manager.

7.1 Set up your Server Manager

1. Browse to the Server Manager at port 8800 on your server IP (e.g. https://12.34.56.78:8800)
   a. If you have configured a DNS entry for your server, you may use that instead of the IP address
2. Accept the initial self-signed certificate and click Continue to Setup, and accept the security warning
3. Enter the Hostname of your server, and:
   a. If you have a CA-issued TLS/SSL certificate file and its corresponding key file, select them here and click Upload & Continue
   b. If you do not have a CA-issued certificate to use, click Use Self-Signed Cert
4. Click Choose License and select the license (.rli) file provided to you by the Plotly sales team
5. Enter a new password for the admin console, then click Continue

7.2 Run preflight checks

These pre-installation checks are designed to ensure that your server meets the requirements to run Dash Enterprise smoothly. Once they have passed, click Continue.

Some check failures or warnings do not impede installation or use of Dash Enterprise (see next subsection). If any checks apart from those result in errors or warnings, consult Plotly support before proceeding.

7.2.1 Preflight check failure: no route to host

Errors similar to dial tcp 10.0.20.51:9879: getsockopt: no route to host occasionally occur on Centos/RHEL systems and are typically a result of firewall restrictions on your server. To determine whether the firewall is the issue:

- Open a terminal and run sudo systemctl stop firewalld
- Reload the preflight checks page

If the above resolves the issue, disable firewalld (if possible) to continue. If disabling firewalld is not an option, add docker0 to the list of active zones, using the below commands as an example:

firewall-cmd --zone=trusted --add-interface=docker0
firewall-cmd --reload
7.3 Configure basic settings

1. Read and accept the **Plotly End User License Agreement**
2. Specify the path to your **Plotly Data Directory**
   - Installation on a Cloud Provider or VM Edition: use the default /plotly
   - Installation on your Own Server: use the data directory created in Section 3.2
   - Fully Offline Installations: use the data directory created in Section 4.2.4
3. (If using Dash Enterprise) Under **Dash Deployment Server**, check Enable Dash On Premise
   - Leave Dash Deployment SSH Port at its default value (3022) unless you are already using this port number for a different purpose
4. (If intending to use streaming functionality) Check Enable Streaming
5. Under **Base Domain Name and SSL certificate**, enter the base domain name you selected in Section 1.1 and select a TLS/SSL certificate option:
   - Use the Server Manager certificate to use the same certificate that you installed while following Section 7.1
   - Automatically generate a self-signed certificate
   - Upload an existing certificate and key to use a separate CA-issued certificate
     ○ The certificate file must contain the full CA chain (including all intermediate certificates) in order to avoid errors when connecting to your On-Premise instance via a browser or the Plotly API libraries
6. If you enabled Streaming: repeat step 5, but use the streaming domain name you chose
7. If you enabled Dash On Premise: repeat step 5, but use the Dash domain name you chose

7.3.1 Authentication settings

Dash Enterprise uses its own locally managed authentication system by default. When using the default authentication system, user accounts and metadata are stored locally, and, if an SMTP server has also been configured, users are able to reset their own passwords. Dash Enterprise also supports integration with your existing LDAP or SAML authentication server/identity provider.

As of Dash Enterprise 4.0.0, e-mail addresses are supported as usernames when using LDAP or SAML authentication, subject to the following conditions:

- The address contains no more than one @
- The @ character must be followed by a domain
- The domain cannot contain underscores (_)

Dash users should log in using their normal e-mail address as the username. However, when sharing Dash apps with other users, a Dash app owner should replace the @ symbol with an underscore (e.g. if sharing with localpart@domain.com, enter localpart_domain.com).
At this time, Dash Enterprise only supports app deployment over SSH (not HTTPS) when using e-mail addresses as usernames.

Regardless of which authentication mode you are using, you must designate an administrator by checking Create admin user, and:

- **If using local authentication**: set the **Admin Username** and **Admin Password** fields
  - You may create additional administrators after logging into the Dash user administration panel as this user (see Section 19: Admin panel reference)
- **If using LDAP or SAML authentication**: specify the name of a user that already exists on your authentication server
  - When logging into Dash Enterprise as an administrator, use the password of the designated administrator account as it was set on your LDAP or SAML server

### 7.3.1.1 LDAP authentication

For **LDAP Authentication** (including Active Directory), select **Use LDAP for authentication** and configure the fields as below.

1. **Server URI**: address of your LDAP server (must begin with `ldap://` or `ldaps://`)
2. **Bind DN**: the full distinguished name of the LDAP account to use for user lookups
   - If your LDAP server does not require authentication, set this to an empty string using a pair of double quote marks (`""`)
3. **Bind Password**: the password for the Bind DN account
   - Use an empty string (as above) if your server does not require authentication
4. **Search DN** (a.k.a. **Base DN**): the DN that Dash Enterprise will use to search for users when authenticating
   - Usually related to the LDAP server’s domain name; e.g. `server.ad.your-company.com` will have a Base DN of "CN=Users,DC=server,DC=ad,DC=your-company,DC=com"
   - To use the Global Catalog Server:
     - **With LDAP**: append `:3268` to your LDAP Server URI (e.g. `ldap://ad.your.company:3268`)
     - **With LDAPS**: append `:3269` to your LDAP Server URI
5. (Optional) **Group DN**: usually the same as the Search DN
   - Dash Enterprise will search this tree for valid user groups
   - May be left blank if not using LDAP group sharing with Dash apps
6. (Optional) **Restrict LDAP login based on group membership**: enable this only if you want members of a specified LDAP group to be able to log in
   - Requires the full DN of the group
   - Advanced group checking (including AND, OR, and NOT operations) is supported
7. (Optional) **Install a Self-Signed TLS/SSL Certificate for LDAP** option if your LDAP server uses a self-signed certificate
Not recommended: You may check Disable LDAP Auth Certificate Checking to bypass this option; however, this exposes your authentication to man-in-the-middle attacks.

8. (Optional) Check Use Custom Search Filter box and specify a new Search Filter or Group Filter if needed.

9. (Optional) Check Restrict Dash App creation based on group membership and enter a group name (the full DN is not required here).

   - Note: If any of the LDAP specifications require the use of a single quote, the quote must be escaped in such a way that it can be parsed via a Python YAML interpreter and exported within a Bash script.
     - Any single quote must be represented as follows: "'" "'
     - Example: CN=User's,DC=ad,DC=plot,DC=ly must be entered as CN=User''''''''''''s,DC=ad,DC=plot,DC=ly

10. (Optional) For testing purposes only, check Enable authentication logs to make authentication logs available via a secret URL.

    - NOT recommended for production use, since authentication logs may contain confidential data.

7.3.1.2 SAML authentication

This section assumes you have access to an already-configured SAML 2.0 identity provider (IdP). For guidance setting up an IdP, see Section 20: Configuring Dash Enterprise to use common SAML IdPs.

To configure Dash Enterprise to use a SAML 2 IdP for authentication, choose Use SAML for authentication and configure as follows:

1. Choose an option for the IdP (remote) metadata from the following:
   a. Enter a URL to the IdP (remote) metadata
   b. Provide a file containing the IdP (remote) metadata
   c. No IdP (remote) metadata is available yet: select this if your IdP requires the Dash Enterprise service provider (SP) metadata before it can generate IdP metadata
      - This is an intermediary step and authentication will not work with it enabled
      - When you have configured your IdP, you must return to the Dash Enterprise Settings and choose one of the other IdP metadata options to proceed

2. Copy the SP (local) metadata URL displayed to use to configure your IdP.
   - If your IdP requires a metadata file instead of a URL, download the file by visiting this URL in your browser after Dash Enterprise has started

3. Use name_id from IdP as Plotly Username: leave checked unless you need to use another attribute from your IdP as the username
   - Username values must be 1 to 31 characters long (inclusive) and may contain only alphanumeric characters plus:
     - _ (underscore)
     - . (period)
To use a different attribute as a username, uncheck this option and enter the attribute name in the field that appears.

4. Choose an SSL certificate option that will be used for the signing certificate and, if you also check Enable encryption when communicating with the IdP, for the encryption certificate as well (Dash Enterprise uses a different certificate set for SAML from what it uses for SSL)
   a. **Automatically generate local certificates and keys for SAML**: Dash Enterprise will generate self-signed certificates and keys the next time it restarts, to be used for SAML signing and encryption
   b. **Provide files containing local certificates and keys**: you will need to upload a signing certificate and key, as well as an encryption certificate and key if you enable encryption

5. Check **Enable Signed AuthnRequests** if your IdP requires this

6. If your IdP uses a self-signed certificate for SSL, choose **Disable SSL certificate validation when communicating with the IdP** to suppress certificate warnings

7. For additional debugging information during setup, enable the **Enable SAML Debugging** option
   - **Not recommended for production** due to the large amount of data generated

8. To restrict app creation to a specific SAML group, enable **Restrict Dash App creation based on group membership** option and enter the name of the group
   - This group name should match the information sent as the “groups” attribute in the IdP’s SAML assertion

11. If desired for testing, you can check **Enable authentication logs** to make authentication logs available via a secret URL
   - **Not recommended for production**, since authentication logs may contain confidential data

### 7.3.2 Dash Enterprise Kubernetes additional required configuration

First, configure your Server Manager Settings as for Dash Enterprise Single Server ([Section 7.3: Basic settings](#)). Once that is complete, continue as below.

#### 7.3.2.1 Configure external services and enable High Availability for Dash

1. Under **External Databases**
   a. Check **Use External Postgres Database** and fill out the fields as below:
      i. **Postgres DB Host**
         - If using AWS: the **Endpoint** of your Postgres instance (from your RDS instance’s details)
         - If using GCP: the IP address of your Postgres instance
         - If using Azure: The “Server name” from the instance page on Azure Portal
      ii. **Postgres DB port**: 5432 (default)
iii. Database User:
   - For AWS and GCP: postgres (default)
   - For Azure: “Admin username” from the instance page on Azure Portal

iv. Database Password: whatever was set for the Database User user upon creation

v. Chart Studio and Authentication Database Name: streambed

vi. Dash Deployment Server Database Name: dash_deployment_server

b. Check **Use External Redis Database** and fill out the fields as below:
   i. Redis Host
      - If using AWS: the Primary Endpoint of your Redis cluster
      - If using GCP: the IP address of your Redis instance
      - If using Azure: The Hostname of the redis instance (from the instance page on Azure Portal)
   ii. Redis Port: 6379 (default)
   iii. Access Key (Optional):
      - For AWS and GCP: Can be left blank
      - For Azure: Primary access key from the “Access Key” section of the instance’s page on Azure Portal
   iv. Celery Redis Database Number: caches data related to Celery and task queues; default 0 (but can be any number from 0-16)
   v. Chart Studio and Authentication Database Number: caches data related to Chart Studio user sessions; default 1 (but can be any number from 0-16)
   vi. Dash Deployment Server Database Number: caches data related to Dash Deployment Server user sessions; default 2 (but can be any number from 0-16)

2. Check **Enable High Availability for Dash**
   a. If this option is not available in your Server Manager Settings, contact Plotly support for help with your license

3. Fill out the fields in that section as follows:
   a. Select **Use Google Kubernetes Engine, Amazon Elastic Kubernetes Service or Azure Kubernetes Service** as appropriate for your situation
   b. Docker Registry URL: the URI for the container registry you created
      i. Only use the base DNS name; do not include / or anything after it
   c. Amazon Elastic Kubernetes Service Settings:
      i. EKS Cluster Name: the Name you input when creating the cluster
         - Retrieve this from the EKS Clusters summary page
      ii. EKS Cluster Region: the Location you selected when creating the cluster
         - Retrieve this in the EKS dashboard by clicking on your cluster name and checking the Cluster ARN value
      iii. **Use internal Network Load Balancer (NLB) for incoming connections**: Enabling this option creates the load balancer on a private subnet so that it is not accessible outside the VPC. Prerequisites:
         - For internal load balancers, your Amazon EKS cluster must be configured to use at least one private subnet in your VPC
- The private subnet should have correct tags as per this guide.

d. Google Kubernetes Engine Settings:
   i. GKE Cluster Name: the Name you input when creating the cluster
   ii. GKE Cluster Zone: the Location you selected when creating the cluster
   iii. This information is available in the GCP console’s GKE Clusters summary

e. Azure Kubernetes Service Settings:
   i. Cluster Name: The Name you input when creating the cluster.
   ii. Resource Group: The Resource Group under which the cluster was created.

f. Max pod lifetime in seconds: the default is fine for most purposes
   i. Configure this to a lower value if you find yourself frequently updating configuration settings and need to flush pods with the stale configuration
      - This can be manually accomplished by scaling deployments to 0 and back up as required for your needs

4. Save the settings, then return to the Dashboard and click Stop, then Start

7.3.2.2 Update base domain DNS record to use the load balancer IP

1. Retrieve the external IP for your loadbalancer
   - AWS: Go to EC2 > Load Balancers, find the load balancer associated with your VPC, and copy its DNS name
   - GCP: Go to Kubernetes Engine > Services & Ingress and locate the Endpoint value for dds-elb-service
   - Azure: Go to Load Balancers, find the load balancer named kubernetes belonging to the resource group and cluster name created earlier and locate the “Public IP Address” from the load balancer’s page on Azure Portal. If there are multiple IP addresses attached to the load balancer, choose the one which has ports: 80, 443, 3022 and 8800 exposed in the rules.

2. In your DNS provider, update your existing Plotly Base Domain Name record to use the load balancer address
   - GCP: update the A record to point to the load balancer’s IP address
   - AWS: update the CNAME record to point to the load balancer’s DNS name

7.4 Configure optional settings (if applicable)

1. To store data externally using Postgres in single-server mode, follow Step 1a in Section 7.3.2.1: Configure external databases

2. To cache session and queueing data externally using Redis in single-server mode, follow Step 1b in Section 7.3.2.1: Configure external databases

3. To use AWS S3 buckets to store Chart Studio data, enable External Storage and fill out the following values:
- AWS Access Key ID: ID of the access key (starting with AKI) used to access S3
- AWS Secret Access Key: Secret of the access key used to access S3
- S3 bucket for storing charts data
- S3 bucket for storing chart images and profile avatars

4. Enable **Outgoing Mail Service (SMTP)** to enable Dash Enterprise to allow self-service resets in local authentication mode and keep users informed about collaborators, likes, or comments on plots
   - Use your own email server settings here
   - If your email server does not require authentication, set the Host User and Host Password fields to empty strings using a pair of double quote marks (""")

5. **Private Mode**: By default, access to Dash Enterprise is restricted to logged-in users only, including files marked “public” or shared via a “share key”. To make “public” files viewable without authentication, uncheck **Enable Private Mode**.
   - We **strongly discourage** this if your server is publicly accessible
   - If not using LDAP or SAML authentication, disabling Private Mode permits self-serve account creation

6. To allow data imports from servers with private or reserved IP addresses such as 10.0.0.1 or 169.254.1.1, enable **Allow Data Imports from Private IP Addresses**
   - This is separate from the Database Connector (Falcon) in Chart Studio, which is not affected by this setting
   - If unsure of whether you need this functionality, leave it unchecked

7. To retrieve tickets from your KDC to be used by Dash Apps, check **Retrieve Kerberos ticket after authentication** and specify your default realm and the KDC’s domain name

8. To customize the amount of time a user can remain logged in to Dash Deployment Server or Chart Studio, specify the time in seconds using in **Session length** option
   - Set to 0 (default) for unlimited session length

9. Enable **Dashboard Customizations** to customize Chart Studio appearance as desired
   - Default Banner Title: The default title to be used when a dashboard is created (plain text)
   - Default Banner Links: A list of links to appear in the banner of newly created dashboards (JSON); for example:

```
[
    { "title": "Title of first link","url": "https://url.of.first/link/" },
    { "title": "Title of second link", "url": "https://url.of.second/link/"
]
```

10. If using Mapbox, check **Enable Mapbox Integration** and specify your **Mapbox Default Access Token** and **Mapbox Atlas Default Style URL**
   - Mapbox is not required to access Dash Enterprise’s built-in mapping functionality

11. Search Memory Usage
12. **Enable Daily Snapshots** to store up to 3 daily snapshots
   - For additional information, see [Section 7.6: Using Snapshots](#) below
13. Enter a **Custom pip index URL** to use instead of the public PyPI server
14. Specify any **Allowed Directories for Mapping** desired to allow a non-admin user to map those directories on the host into Dash apps
   - All other restrictions still apply
15. **White-Labeling (Customize Branding):**
   - You can remove the link from Portal to App Manager by disabling the **Display a Link to the App Manager in Portal** option
     i. The link will be removed for both admin and non-admin users
     ii. The App Manager will still remain accessible through URL
   - You can remove Plotly branding from all login prompts by selecting the **Remove Plotly Logo from Login Window** option
16. **Enable Dash Customizations** for the following customization options:
   - Dash Proxy Configuration
   - Dash Web Server Workers
   - Dash Web Server Threads per Worker
   - Indirect Certificate Mapping
     i. Removes a direct volume map from the host's
        `/etc/ssl/certs/ca-certificates.crt` file to the Dash application's
        `/etc/ssl/certs/ca-certificates.crt` file
     ii. This allows for locks to be removed that would otherwise cause failures when deploying apps with dependencies that manage their own certificate updates
17. **To Enable Custom Fonts:**
   - SSH into your server and add the following files to `/plotly/custom_fonts`
     i. Any custom font file you want (in TTF format)
     ii. A `custom_fonts.json` file containing metadata about the fonts in the following format:

```
{
  "fonts": [
    {
      "value": "Fontname MS", "label": "Custom Font"},
    {
      "value": "OtherFont MS", "label": "Custom Font"
    }
  ]
}
```
   - For each font, fill out the `value` field with the `font-family` specified in its TTF file
   - The `label` is arbitrary; this is the name Chart Studio Enterprise will display the font as
iii. A custom\_fonts.css file containing information about the font-family name and location of each TTF file, as follows:

```css
@font-face {font-family: 'Fontname MS'; src: url('/static/webapp/custom_fonts/otherfont.ttf') format('truetype')}

@font-face {font-family: 'OtherFont MS'; src: url('/static/webapp/custom_fonts/otherfont.ttf') format('truetype')}
```

- All URLs must use static/webapp as the root

### 7.5 Other options and customizations

These custom options are intended for very specific customers and situations. Only if you have been advised to do so by the Plotly team, select **Enable Plotly Customizations** or **Enable Docker Customizations** and enter any provided code. Otherwise, leave this option unchecked.

#### 7.5.1 Customize the React Chart Editor (RCE)

Dash Enterprise now supports allowing non-admin users to disable/enable RCE controls based on plotly.js full attribute paths or control names. To use this feature:

1. Create a custom\_config.json file containing a visibility\_rules key, which in turn contains one of the following keys: **blacklist** or **whitelist**
   - **Blacklist**: specify regex rules to run against a plotly.js attribute string or control name and exclude matches from the editor
     - If a plotly.js full path attribute string matches any of the rules in the blacklist array, the corresponding control will **not show up** in the editor
     - You can add exceptions to your rule, as well as exceptions to exceptions
     - **Example** custom\_config.json file with a blacklist rule:

```json
{
  "visibility\_rules": {
    "blacklist": [
      {
        "type": "attrName",
        "regex\_match": "color",
        "exceptions": [
          {
            "type": "attrName",
            "regex\_match": "other\_color"
          }
        ]
      }
    ]
  }
}
```
The above:
- Hides all controls with the `color` string in their plotly.js full attribute path
- Does not hide controls with `marker.color` in their full attribute path (an exception to the blacklist rule)
- Hides `ColorPicker` controls (an exception to the blacklist rule’s exception)

**Whitelist:** similar logic applies as above; this rule lists the only attributes that should show, as opposed to the attributes that should be hidden

- Example `custom_config.json` file with a whitelisting rule:

```json
{
  "visibility_rules": {
    "whitelist": [
      {
        "type": "attrName",
        "regex_match": "color",
        "exceptions": [
          {
            "type": "attrName",
            "regex_match": "marker.color",
            "exceptions": [
              {
                "type": "controlType",
                "regex_match": "ColorPicker"
              }
            ]
          }
        ]
      }
    ]
  }
}
```
The above will:

- **Show** all controls with that have the `color` string in their plotly.js full attribute path
- **Hide** the ones that have `marker.color` in their full attribute path
- Show ColorPicker controls even if they have the `marker.color` string in their attribute path

**Exceptions/rules format:** both exceptions and rules are objects that must have a `type` key and a `regex_match` key

- The possible `type` keys are:
  - `attrName` (if you want to add a rule or exception based on a plotly.js attribute path)
  - `controlType` (if you want to add a rule based on an RCE control Type).

- The `regex_match` key should be a regular expression string to compare against your plotly.js attribute paths or component displayNames

- All valid regex strings are supported

- If either the `type` or `regex_match` key are omitted in your rule or exception, your custom_config.json file will be considered invalid and no rules will be applied

2. Ensure your file is properly formatted as JSON and copy it into a `/plotly/rce` directory on your server (create it if it does not already exist)

3. In the Server Manager Settings, go to the Special Options & Customizations section and check **Enable custom Chart Studio configurations**

When this is complete, your custom rules will be applied when using the chart editor as a non-admin user.

# 7.6 Using Snapshots

Dash Enterprise has improved support for Snapshots, which are point-in-time backups initiated via the Server Manager and intended to be used as restore points when completing a fresh installation on a new server.
Snapshot duration takes approximately 10 minutes on a server using a solid-state drive and writing to its own local filesystem (*not recommended for production systems*). You may achieve further performance improvements if your Dash Enterprise instance is on AWS EC2 and you set up an AWS S3 storage bucket as the snapshot file destination.

**Configuring snapshots**

1. Browse to your Dash Enterprise Server Manager and click **Settings**
2. Under **Daily Snapshots**, check **Enable Snapshots**
3. Click **Save**
4. Click the gear icon in the top right, then **Console Settings** in the resulting drop-down menu
5. Under **Snapshot & Restore**, configure your backup strategy according to your business needs; we recommend the following:
   a. **Snapshot File Destination**: avoid using Local for production systems
   b. **Snapshot Timeout (optional)**: set to 30 minutes
6. Click **Save**

**Taking a snapshot**

There are two options for taking snapshots:

- **Automatically**: in the Server Manager Console Settings, check **Enable Automatic Scheduled Snapshots** and select your preferred frequency
- **Manually**: on the Server Manager Dashboard, click **Start Snapshot**

While a snapshot is in progress, you may continue using Dash Enterprise as normal.

If a snapshot is interrupted or otherwise does not complete, you may retry manually with **Start Snapshot**. Automatic scheduled snapshots do not retry automatically upon failure and require a manual retry.

**Restoring from snapshot**

Dash Enterprise only supports restoring snapshots during installation **on a new server**. Restoring in place with an existing installation is not supported.

1. SSH to your target server and run the Plotly installation script:
curl -sSSL https://get.plot.ly | sudo bash

2. Browse to the IP address generated by the script and click **Continue to Setup**
3. Accept the self-signed certificate and proceed to the next page
4. Enter the **Hostname** of your server and upload the key/certificate files you wish to use, or proceed with a self-signed certificate
5. At the Upload Your License page, click **Restore from Snapshot**
6. Enter the location of your snapshot file and click **Browse Snapshots**
7. When your snapshot list appears, click **Restore** beside the snapshot you want to restore and wait for the snapshot metadata to load
8. When prompted, unlock the admin console and proceed past the Preflight Checks
9. At Restore Cluster, click **Restore**, then monitor progress from the Server Manager Dashboard
10. When the app reports it is ready, SSH into your server and rebuild your Dash apps:

   ```bash
   docker exec -it dash bash
   for service in $(dokku --quiet redis:list); do dokku redis:start $service; done
   for service in $(dokku --quiet postgres:list); do dokku postgres:start $service; done
   dokku ps:rebuildall
   ```

Your restored Dash Enterprise instance is now ready for use.

### 7.7 Running and logging into Dash Enterprise

When you are satisfied with your configuration, click **Save** at the bottom of the page, then **Apply Now** to restart Dash Enterprise. Startup progress can be monitored from the Server Manager dashboard.

To log in to Dash Enterprise for the first time, browse to your base domain URL and authenticate, either with the administrator credentials you set in the Server Manager Settings (if using local authentication) or existing credentials stored on your LDAP or SAML server.

### 7.8 Next steps

Congratulations! Dash Enterprise is now ready for use.

If using Plotly’s API client libraries (Python, R, or MATLAB), proceed to configure them to access Dash Enterprise instead of the public Plotly Cloud service (see [Section 9: Client library setup](#)).

We recommend checking for updated versions of Dash Enterprise regularly (See [Section 8: Upgrade Dash Enterprise](#)).

Finally, if you run into any issues using Dash Enterprise, consult [Section 10: Troubleshooting and problem solving](#).
Enjoy using Dash Enterprise!
8. Upgrade Dash Enterprise

This section of the documentation deals with upgrades of existing Dash Enterprise installations. If you have not yet installed Dash Enterprise, refer back to sections 2 through 6 earlier in this document.

These instructions assume that your current Dash Enterprise instance is at least version 3.4.0. You must upgrade to 3.4.x before upgrading to 4.0.1; it is not possible to skip major versions when upgrading.

8.1 Prepare for upgrade

Before upgrading Dash Enterprise, stop your existing instance and back up your data, as follows:

1. Browse to the Server Manager dashboard on port 8800 and click Stop Now
2. Back up your Dash Enterprise data directory!
   a. Check data size: To determine how much disk space your data directory is using, run the command in this example:

      
      $ sudo du -sh /plotly
      540M /plotly

      ● Replace /plotly with the name of your data directory if you used a different name during installation
      ● The command may take a while to complete depending on data size; one minute is typical
      ● In this example, the data directory is using 540 megabytes
   b. Check disk space: make sure select use ldap you have enough free space for the upgrade
      ● Check the space in both the Docker data directory (default /var/lib/docker) and the Plotly data directory (/plotly)
      ● We recommend at least 80 GB for the Docker data directory (/var/lib/docker) and at least 200 GB for /plotly before proceeding.
      ● If using Dash and you require Python packages to be installed along with your application, we suggest at least 100 GB for /var/lib/docker
      ● To check how much disk space you have free on your current machine, use:

      
      $ sudo df -h /var/lib/docker
      $ sudo df -h /plotly

      The output should look something like this:

      Filesystem Size Used Avail Use% Mounted on


In this example, there are 390 GB of space available — enough space to hold a backup copy of the data directory during the upgrade process.

c. **Copy the data directory:** to back up the data directory, copy it to a new location:

```bash
$ sudo cp -a /plotly /plotly.bak
```

3. **Upgrade Docker** if it is older than version 17.05.0 (minimum version supported by Dash Enterprise 4.0.1) as follows:
   a. If your server can access the Internet, re-run the installation script per Step 3 in Section 3.2: Installation using Plotly Script
   b. For an airgapped installation, download the appropriate Docker package and any required dependencies to a storage device, then transfer the package to your server and install it with:
      - **CentOS/RHEL/Fedora:** `sudo rpm -ivh <package_name>.rpm`
      - **Ubuntu / Debian:** `sudo dpkg --install <package_name>.deb`

## 8.2 Upgrade

Upgrade steps vary depending on the type of installation you are using.

### 8.2.1 Installation on cloud provider or installation on own server

This process applies to both Dash Enterprise Single Server and Dash Enterprise Kubernetes.

1. Browse to the Dash Enterprise Server Manager at port 8800 of your base domain
   ○ You will see a notice that there is an update available
2. Select **View Update** to see a list of available updates, then click **Install Update**.

If you need to change channels to upgrade, consult Section 17: Change channels for an upgrade.

### 8.2.2 Fully offline (airgapped) installation

1. Download a new version of Dash Enterprise and rename the file to add the `.airgap` extension
   ○ If there is more than one new version between your current version and the one you are upgrading to, contact support as you will also need to download certain intermediate versions and they can instruct you which ones are necessary.
2. Check the changelogs to see if Replicated has been updated by the release you’re upgrading to (or any intermediate release you will be installing).
If so, on an internet capable machine, download the latest version mentioned using https://s3.amazonaws.com/replicated-airgap-work/stable/replicated-2.45.0%2B2.45.0%2B2.45.0.tar.gz (replace all 2.45.0 with the version you will need)

Upload the resulting file to your server and extract it using `tar xzvf replicated-2.45+2.45+2.45.tar.gz` (modify this file name as needed)

Run `cat ./install.sh | sudo bash -s airgap`

3. Browse to the Server Manager at port 8800 and click the gear icon in the top right to access the Console Settings
4. Find the Airgapped Settings section and enter the paths to the new .airgap and your license file
5. Save your changes
6. Go back to the Dashboard; you should see a notice that there is an update available
7. Select View Update to see a list of available updates, then click Install Update

8.2.3 VM Edition

Please contact Plotly support for help upgrading the VM Edition.

8.3 Configure the upgraded instance

After the upgrade has been installed, browse to the Settings page of Dash Enterprise Server Manager and confirm your server is configured as desired. For more information, see Section 7.3: Basic settings.

8.4 Run the upgraded instance

After checking your configuration, click Save to start the new version. You can monitor startup progress on the Server Manager dashboard.

8.5 Online migrations

Chart Studio Enterprise automatically performs migrations to upgrade data from one version to the next. Online migrations start 5 minutes after the server boots. You may stop or reboot the server while online migrations are running and they will resume automatically. To see the status of online migrations, browse to https://plotly.your.company/online_migration_status. These migrations do not apply to Dash Enterprise and this status url will only be accessible when running Chart Studio Enterprise.

Current migrations and their impact:

- **Fix invalid dashboards**: Fixes any invalid dashboards so they can be displayed by Chart Studio. Until this operation completes, any invalid dashboards may not be displayed. (In previous versions, errors due to invalid dashboards were ignored.)
8.6 Clean up after the upgrade

Once you are confident that the upgrade was successful, you may remove your backup with the following command:

```
$ sudo rm -rf /plotly.bak
```

There is no disadvantage associated with keeping the backup file as long as you have enough disk space.
9. Client library setup

If you are sending data to Dash Enterprise using one of the Plotly API client libraries, you must configure it to use Dash Enterprise instead of Chart Studio Cloud. Each user will still need to generate credentials, and each user’s Dash Enterprise API key will be different from the Chart Studio Cloud API key.

9.1 Python

See our online documentation for more details.

1. Install the latest version of the library with:
   ```bash
   pip install --upgrade plotly
   ```

2. Set the endpoint of your server by running the following script:
   ```python
   import plotly.tools as tls
   tls.set_config_file(plotly_domain='https://plotly.your-company.com',
                      plotly_streaming_domain='stream.plotly.your-company.com,
                      plotly_api_domain='plotly.your-company.com')
   ```

   - Replace `plotly.your-company.com` with the domain names you set up when configuring Dash Enterprise
   - The script saves these endpoints as a JSON file in `~/.plotly/.config`

9.2 R

1. Install the latest release from CRAN with:
   ```r
   install.packages("Plotly")
   ```

2. If you have already set the endpoint (for instance, when using Python or MATLAB), there is no need to set it again. Otherwise, set the endpoint in R as follows:
   ```r
   set_config_file("https://plotly.your-company.com",
                   "https://stream.plotly.your-company.com")
   ```

   Replace `plotly.your-company.com` with your Base Domain Name.

See our online documentation for more details.

9.3 MATLAB

1. If you have already set the endpoint (for example, for Python or R), there is no need to set it again. Otherwise, set the endpoint in MATLAB as follows:
   ```matlab
   >> saveplotlyconfig("https://plotly.your-company.com",
                       "https://stream.plotly.your-company.com")
   ```
9.4 Other libraries/plugins

Plotly also supports other libraries/plugins for Node.js, Arduino, Julia, and Excel. For more information on how to configure these endpoints, contact Plotly support.

See our online documentation for more details.
10. General troubleshooting and problem solving

10.1 LDAP: Testing authentication

You can troubleshoot LDAP authentication by running the `ldapsearch` tool on your Dash Enterprise server. If you installed Dash Enterprise following the Installation on a Cloud Provider instructions, this tool has already been installed, otherwise you can install it as below:

- **Ubuntu**: `sudo apt install ldap-utils`
- **RHEL or CentOS**: `yum install openldap-clients`

`ldapsearch` syntax can be complicated, so here is an example to get you started:

```
$ ldapsearch -H ldap://ad.plot.ly -x -D 'plotly\mrplot' -w 'ldappass' -b 'CN=Users,DC=ad,DC=plot,DC=ly' SAMAccountName -v
```

Breaking it down:

- **ldap://ad.plot.ly** is the server URI
  - This is the auth_ldap_server_uri parameter
- **plotly\mrplot** is the bind DN used for authentication
  - This is the auth_ldap_bind_dn parameter
  - **mrplot** is the username
  - Your LDAP server may or may not need you to include the LDAP domain and backlash (`\plotly\` in this example); check with your server administrator
- **ldappass** is the password for the bind DN
  - This is the auth_ldap_bind_password parameter
- **CN=Users,DC=ad,DC=plot,DC=ly** is the search DN
  - This is the auth_ldap_search_dn parameter
  - This is usually based on the domain name of your LDAP server

The above command returns a list of all accounts in the search DN; in this case, the list of all usernames allowed to log in to Dash Enterprise.

If you receive any other output, contact your LDAP server administrator or Plotly support for help.
10.2 Testing without adding entries to your DNS server (using /etc/hosts)

You can test your Dash Enterprise instance locally by adding the DNS entries directly to the hosts file on each machine intended to access the instance. Here are instructions on how to do that for various operating systems. The line to add will look something like this:

```
12.34.56.78 plotly.your-company.com plotly-stream.your-company.com
```

- Replace 12.34.56.78 with the IP of your Dash Enterprise server
- plotly.your-company.com plotly-stream.your-company.com are the Base Domain and Streaming Domain values, respectively, that you specified during configuration

If you set your server’s current hostname as the Base Domain in your Dash Enterprise configuration, and you do not plan to use streaming, you do not need to modify your hosts file.

10.3 SSL Error: certificate verify failed when using the Plotly Python library

These kinds of errors look like this in your console:

```
SSLError: [Errno 1] _ssl.c:510:
error:14090086:SSL routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify failed
```

You can solve this in one of two ways:

- Set the path to a valid certificate file in your Plotly Python configuration (recommended):

  ```
  >>> import plotly.tools as tls
  >>> tls.set_config_file(plotly_ssl_verification='/absolute/path/to/file.crt')
  ```

- Turn off SSL verification in your Plotly Python configuration (not recommended):

  ```
  >>> tls.set_config_file(plotly_ssl_verification=False)
  ```

Only use this option if you are sure the endpoint of the HTTPS request points to your server. This option leaves your data vulnerable to man-in-the-middle attacks.
When using IPython, if you modify `plotly_ssl_verification` as above, you may receive formatter warnings similar to:

```
Exception image/{format} formatter: [Errno 1] _ssl.c:510: error:14090086:SSL routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify failed
```

These warnings can safely be suppressed as follows:

```python
>>> import warnings
>>> import IPython.core.formatters
>>> warnings.filterwarnings('ignore', category=IPython.core.formatters.FormatterWarning)
```

All other warnings will continue to be displayed.

### 10.4 Private key/certificate debugging

If you suspect your SSL certificate and key do not match, you can verify by finding the `sha256sum` of each as follows:

```
openssl x509 -in /path/to/your/plotly_certificate.crt -pubkey -noout -outform pem | sha256sum
```

```
openssl pkey -in /path/to/your/plotly_key.key -pubout -outform pem | sha256sum
```

If the results of the above commands match, then the key and certificate match as well.

### 10.5 Running `devicemapper` in loopback mode

If your system defaults to using `devicemapper` in loopback mode (typical for RHEL 7 and CentOS 7), you will see an error during installation that looks like this:

The running Docker daemon is configured to use the 'devicemapper' storage driver in loopback mode.
This is not recommended for production use. Please see the following URL for more information.


Do you want to proceed anyway?

To solve this, cancel installation and configure your system to use `direct-lvm` according to [this Docker guide](https://help.replicated.com/docs/kb/developer-resources/devicemapper-warning/).
10.5 Dash app manager boot failure -- Error response from daemon: max depth exceeded

During startup, you may encounter an error that corresponds with the following message in your Dash container logs:

```
Error response from daemon: max depth exceeded
```

You can resolve this as follows:

1. Find the Dash container ID: `sudo docker ps`
2. Stop the Dash container: `sudo docker stop DASH_CONTAINER_ID`
3. Delete all Dash-related containers/images:
   ```
sudo docker rm -f $(sudo docker ps -a | grep -E "web.1 | Exited" | awk "\{print $ 1\}"

sudo docker rmi -f $(sudo docker images -a | grep -E "<none> | dokku" | awk "\{print $ 3\}"
   ```
4. In the Dash Enterprise Server Manager, click **Start Now**
5. Redeploy your Dash application

10.6 Plotly data directory verification failure

If attempting to set the Plotly data directory and the verification test is failing:

1. Confirm the directory exists on the system and you have correctly entered the path to it
2. Ensure SELinux is enabled and running with `sudo sestatus`, and enable it if not
3. If SELinux is enabled, add SELinux permissions to your data directory with:
   ```
sudo chcon -Rt svirt_sandbox_file_t /path/to/your_data_directory
   ```

10.7 Missing **container-selinux** dependency (Error: nothing to do)

In RHEL, the **container-selinux** dependency may be missing from the system, which commonly results in the following error when running the installation script:

```
Error: nothing to do.
```

To solve this, we recommend the following options:
Enable the appropriate repository:

```
sudo subscription-manager repos --enable=rhel-7-server-extras-rpms
```

If the above doesn’t work, install the package directly:

```
yum install http://mirror.centos.org/centos/7/extras/x86_64/Packages/container-selinux-2.36-1.gitff95335.el7.noarch.rpm
```

10.8 OCI runtime error

On RHEL < v7.4, OCI runtime issues have been known to occur, preventing Server Manager containers (names beginning with `replicated` from starting when using Docker >= 17.12.1. You can resolve this by upgrading to >= RHEL v7.4.

10.9 Dash Enterprise Kubernetes setup problems

If you followed the steps in Section 6.2: Set up your cloud infrastructure and 7.3.2 Dash Enterprise Kubernetes additional required configuration and On-Premise does not start up correctly, here are some troubleshooting steps to try:

1. In your cloud provider’s console, confirm the following:
   - All of your cluster resources were created on the same VPC network
   - The VPC has the appropriate firewall rules/security groups to allow traffic between your resources
   - Your service accounts/IAM roles were created with the appropriate permission sets
   - Your EC2/compute instances and Kubernetes clusters were created with the appropriate role account assigned to them

2. Check for any typos in the values configured in your Server Manager Settings

3. `ssh` into the EC2 or Google Compute Engine instance you created in Step 3 and check the `docker logs` output for the `streambed` and `dash` containers

4. Inspect your pod logs with `kubectl logs` for clues

5. To manually apply a recent configuration change, manually `scale your deployments` to 0 and back to their original number again

10.9 Support

Our support team is available to help if you run into any issues with your installation. When contacting us, please also include a support bundle, which is a collection of log files and app state that will help us diagnose your problem. There are two methods available for creating and sending one:
10.9.1 Directly upload support bundle to our portal (recommended)

1. SSH into the server running Dash Enterprise
2. Copy and paste the following command, which generates a bundle and prompts you to upload it to our support portal:

```
docker pull replicated/support-bundle && 
docker run --rm -it --name support-bundle 
  -v $PWD:/out -w /out 
  -v /var/run/docker.sock:/var/run/docker.sock 
  --net host --pid host 
  -e HTTP_PROXY -e HTTPS_PROXY -e NO_PROXY 
  replicated/support-bundle generate 
  --channel-id 2981105c67e98b19d315128c52286160
```

3. E-mail Plotly support with a description of the issue and let us know you have made a support bundle available for analysis

10.9.2 Download and e-mail support bundle (for airgapped instances)

1. Browse to your Dash Enterprise Server Manager
2. Click the Support tab
3. Click Download Support Bundle
4. Upload the resulting file to a cloud storage service of your choice
5. E-mail us with a description of the issue and a link to the support bundle
11. Load balancer configuration

These instructions assume you are using a Classic Elastic Load Balancer (ELB) on Amazon Web Services. To configure a different type of load balancer, please consult the documentation for your load balancer.

11.1 Initial considerations

If using Dash Enterprise Single Server behind a load balancer, the ELB will need to be configured to pass requests to the server.

11.2 Port setup

The following ports must be opened on the ELB and the ELB target instance:

- 443 (HTTPS)
- 80 (HTTP)
  - This port is optional, as all requests are made over HTTPS
  - Enabling port 80 will allow Dash Enterprise to automatically redirect requests made on HTTP to HTTPS
- 8800 (Server Manager)
- 3022 (SSH connections to Dash Enterprise)
  - You cannot use standard SSH port 22 for this unless you disable SSH on the server, which we do not recommend

11.3 Listener setup

Configure listeners on ports 443, 80, and 3022 to point to instance ports 443, 80, and 3022, respectively.

If using streaming over SSL, the Server Name Indicator (SNI) is a required component. For the front-end and back-end connections, use TCP listeners on port 443. The load balancer passes requests through along with the SNI certificate, and you can then handle HTTPS termination within the EC2 instance.

11.4 Domain (DNS) setup

Domain entries should be set as CNAMEs pointing to the ELB DNS Name. For example, if the ELB has a DNS name of test.us-east-1.elb.amazonaws.com, DNS entries should be added as follows:

plotly.your-company.com CNAME for test.us-east-1.elb.amazonaws.com
dash.your-company.com CNAME for test.us-east-1.elb.amazonaws.com
If using streaming, also add:

plotly-stream.your-company.com   CNAME for test.us-east-1.elb.amazonaws.com

Then specify the domains in the Server Manager Settings as usual:

- **Base Domain:** plotly.your-company.com
- **Dash Deployment Server Domain:** dash.your-company.com
- **Streaming Domain:** plotly-stream.your-company.com
12. Reverse proxy setup instructions

The following instructions walk you through setting up an Apache reverse proxy to establish an HTTP basic authentication layer before the application. These steps are specific to Ubuntu 14.04 or higher but are generic enough to apply to other reverse proxies.

12.1 Overview

The above diagram outlines the server layout for running Dash Enterprise behind a reverse proxy. The reverse proxy receives requests from the client intended for the host, then passes authenticated requests to the authentication server (Streambed) listening on the internal IP used by Docker.

host_addr = ip address of host server
docker_addr = internal ip address used by Docker
A typical request works as follows:

1. The user's machine performs a DNS lookup for `plotly.your.domain` and gets the IP address of the server
   a. `plotly.your-company.com` is the Base Domain Name configured in the server settings
2. The user's machine connects to this IP via client and reaches the reverse proxy
3. The reverse proxy then sends an authenticated request to `plotly.your.domain`
4. The request is forwarded to the Dash Enterprise instance

Step 4 typically requires adding an entry to the `/etc/hosts` file on the host server to allow the reverse proxy's request for `plotly.your.company` to be directed to the internal IP used by Docker. This is further discussed in Section 12.4: Proxy Configuration.

### 12.2 Prepare for reverse proxy

Once Dash Enterprise has been installed, enable **Local Proxy Mode** as follows:

1. Browse to the Dash Enterprise Server Manager Settings page
2. Check **Enable Dash Customizations**
3. Check **Enable Local Proxy Mode**
4. Click **Save** at the bottom of the screen, then click **Restart Now** to apply your changes

As a result of this change, Dash Enterprise will use Docker’s internal IP address, freeing up the host server’s IP address for use by the reverse proxy. The application will not respond to web requests until the reverse proxy is running.

### 12.3 Install the proxy

After Dash Enterprise has been set to use Local Proxy Mode, install the proxy as follows:

1. SSH into your server
2. Install Apache ([guide](#))
3. Install the following Apache modules:
   ```
   a2enmod proxy
   a2enmod proxy_http
   a2enmod headers
   a2enmod ssl
   ```
   And, if using LDAP authentication, also install:
12.4 Configure the proxy

Modify the configuration files associated with the web server to establish the authentication layer and reverse proxy, as follows.

12.4.1 Update the virtual host file

Replace the contents of the virtual host file (etc/apache2/sites-enabled/000-default.conf by default) with the following:

```
<VirtualHost *:80>
    ServerName plotly.your.domain
    Redirect / https://plotly.your.domain/
</VirtualHost>

<VirtualHost *:443>
    ServerName plotly.your.domain
    ProxyPass / https://plotly.your.domain/
    RequestHeader unset Authorization

    SSLProxyEngine on
    SSLEngine On
    SSLCertificateFile /etc/apache2/server.crt
    SSLCertificateKeyFile /etc/apache2/server.key
    SSLCertificateChainFile /etc/apache2/ca.crt
</VirtualHost>
```

12.4.2 Configure the authentication layer

These instructions only cover local/LDAP HTTP basic authentication since the Plotly API clients (Python, R, or MATLAB) currently only support reverse proxies that require HTTP basic authentication. Other authentication implementations are also possible depending on your reverse proxy’s capabilities.

1. Edit 000-default.conf to add the following Location directives in the <VirtualHost *:443> section:

```
<Location "/clientresp"
    Satisfy Any
    Allow from all
</Location>
```
● This allows access to two required Dash Enterprise endpoints: /clientresp and /__internal/ping
● 000-default.conf should look similar to the below example from the Plotly test site:

```xml
<VirtualHost *:80>
  ServerName plotly.test.com
  Redirect / https://plotly.test.com/
</VirtualHost>

<VirtualHost *:443>
  ServerName plotly.test.com
  RequestHeader unset Authorization
  SSLProxyEngine on

  SSLEngine On
  SSLCertificateFile /etc/apache2/server.crt
  SSLCertificateKeyFile /etc/apache2/server.key
  SSLCertificateChainFile /etc/apache2/ca.crt

  <Location />
    AuthType Basic
    AuthName "Plotly Test Authentication Layer"
    AuthUserFile /etc/users
    require valid-user
  </Location>

  <Location="/clientresp">
    Satisfy Any
    Allow from all
  </Location>

  <Location="/__internal/ping">
    Satisfy Any
    Allow from all
  </Location>
</VirtualHost>
```

2. Use the `htpasswd` utility to create credentials for each new user you want to have access, which will be stored in /etc/users:
3. Confirm the IP address of your host server (the `host_addr` parameter):

   ```
   hostname -i
   ```

3. Modify the `Listen` directives in `/etc/apache2/ports.conf` to allow the server to accept incoming requests only on the `host_addr` address. The configuration file should look similar to:

   ```
   Listen host_addr:80
   
   <IfModule ssl_module>
       Listen host_addr:443
   </IfModule>
   
   <IfModule mod_gnutls.c>
       Listen host_addr:443
   </IfModule>
   ```

4. Next, find the internal IP address used by Docker (`docker0_addr`):

   ```
   ifconfig docker0 | grep -oP 'inet addr:\K\S+'
   ```

5. Edit `/etc/hosts` to include an entry for the `docker_addr` that maps to the Base Domain and Streaming Domain associated with your Dash Enterprise instance:

   ```
   10.1.2.3 plotly.your-company.com plotly-stream.your-company.com
   ```

   - Replace `10.1.2.3` with the `docker0_addr` you discovered in Step 4
   - Apache will now connect to the authentication server (using the Docker address) when it connects to `plotly.your-company.com` or `plotly-stream.your-company.com`
   - Connections coming from outside the server will go to Apache as usual

6. Restart the Apache service for the changes to take effect

### 12.5 Run and log in

   - You should see an authentication prompt similar to:
2. Enter one of the sets of credentials that you configured using `htpasswd` to access the Dash app manager.

3. Proceed to create a new account using the **Sign Up** button in the top right.
   - If using LDAP, skip this and use your existing LDAP credentials to log in.
13. Docker storage driver requirements and considerations

Docker images consist of a series of read-only layers, each representing an instruction in the image's Dockerfile. When Docker creates a container from an image, it creates a new writable layer (container layer) on top of the underlying image layers. The storage driver configured in Docker determines how your host system interacts with these layers.

Typical Docker setups will automatically configure the most appropriate storage driver for a given system. You can determine which storage driver is being used by running `docker info` on the host machine, which will return output similar to:

```
$ docker info
Containers: 0
Images: 0
Storage Driver: overlay
Backer Filesystem: extfs
...
```

In this example, Docker is using the `overlay` storage driver.

A storage driver sometimes needs to be explicitly configured for a given system, depending on the following:

- The host’s operating system, distribution, and Docker edition
- The backing filesystem (i.e. where `/var/lib/docker` is located)

### 13.1 Operating system, distribution, and Docker edition

`overlay2` is the recommended storage driver whenever possible, but any of the below should work:

- **Ubuntu**: `aufs, devicemapper, overlay2 (Ubuntu 14.04.4 or later, 16.04 or later), overlay, zfs, vfs`
- **Debian**: `aufs, devicemapper, overlay2 (Debian Stretch), overlay, vfs`
- **CentOS**: `devicemapper, vfs`
- **Fedora**: `devicemapper, overlay2 (Fedora 26 or later, experimental), overlay (experimental), vfs`

Please contact Plotly support if you need help configuring a storage driver while using a Docker edition other than Docker CE.
13.2 Backing filesystem (where `/var/lib/docker` is located)

Specific backing filesystems are required for some storage drivers:

<table>
<thead>
<tr>
<th>Storage driver</th>
<th>Supported backing filesystems</th>
</tr>
</thead>
<tbody>
<tr>
<td>overlay, overlay2</td>
<td>ext4, xfs</td>
</tr>
<tr>
<td>aufs</td>
<td>ext4, xfs</td>
</tr>
<tr>
<td>devicemapper</td>
<td>direct-lvm</td>
</tr>
<tr>
<td>btrfs</td>
<td>btrfs</td>
</tr>
<tr>
<td>zfs</td>
<td>zfs</td>
</tr>
</tbody>
</table>

Additionally, some backing filesystems will require additional configuration depending on the host system’s OS and kernel version, as discussed in Section 13.3: Further considerations and cautions.

13.3 Further considerations and cautions

Running the devicemapper storage driver in loopback mode is highly unstable and not recommended for production systems. Instead, configure devicemapper in direct-lvm mode as described here. (Note that devicemapper is the default storage driver for Centos 7/RHEL 7 systems.)

Occasionally, storage drivers may have additional prerequisites that could render them incompatible with your system or require additional system configuration. This includes kernel version requirements as well as filesystem formatting. For example, the xfs backing filesystem is supported for the overlay storage driver, as long as `d_type=true` is set.

See the Docker storage driver documentation for additional information on storage drivers and selection/configuration.
14. Import an LDAPS certificate from Active Directory into Dash Enterprise

1. On your Active Directory domain controller, export the certificate for your domain controller as a Base-64 encoded X.509 (.CER) file and save it to your workstation
   - See the Microsoft documentation for details
2. Browse to your Server Manager Settings
3. Select the Use LDAP for authentication radio button and check Install a Self-Signed SSL Certificate for LDAP
4. Click Choose File and select the certificate file you just exported, and click OK
5. Click Save at the bottom of the page and Restart now to apply this change
15. Deploy Dash apps via HTTPS

Dash Enterprise supports deploying Dash apps over HTTPS, which is useful when SSH-based Dash app deployments are not an option, such as when running behind a restricted company proxy.

The process is nearly identical to SSH-based deployments, with the following exceptions:

- You do not need to upload SSH keys to your Dash Enterprise server manager
- You do not need to modify ~/.ssh/config
- When deploying your app, adding git remote looks like:

  ```
git remote add plotly https://plotly.your-company.com/GIT/my-first-dash-app
  ```

Once you have initialized your app in the Dash App Manager, you can deploy code using the same process as with SSH-based deployment:

```
$ git status  # view the changed files
$ git diff   # view the actual changed lines of code
$ git add .  # add all the changes
$ git commit -m 'a description of the changes'
$ git push plotly master
```

After running git push, you will be prompted for a username and password; enter the credentials of the Dash app’s owner (specified in the app’s config.py file).
16. Transfer Dash Enterprise to a new server

1. Browse to the Server Manager and click Stop Now to stop the running instance
2. Back up your Plotly data directory (see Step 2 in Section 8.1: Prepare for upgrade for guidance)
3. Transfer your data directory to your new server using your preferred method (e.g. rsync)
4. Install Dash Enterprise from scratch on your new server using the Plotly instructions earlier in this guide that are appropriate to your use case
   a. When configuring the new instance’s Settings, specify the transferred data directory as your new Plotly data directory
   b. You will need to SSH into your server and rebuild your Dash services and apps from within the dash container for them to start up correctly:

   ```
docker exec -it dash bash

for service in $(dokku --quiet redis:list); do redis:start $service; done

for service in $(dokku --quiet postgres:list); do postgres:start $service; done

dokku ps:rebuildall
```
5. Once the installation is complete and running on your new server, you may stop the running instance on your old server
17. Sync license changes

If Plotly support has made changes to your license, a sync is required for these changes to be reflected in your Dash Enterprise instance:

1. Browse to the Dash Enterprise Server Manager
2. Click the gear icon in the top right, then click View License
3. Select Sync License
18. Change channels for an upgrade

Typically you will only need to change to use a special release. To do this:

1. Browse to the Server Manager and click the gear icon in the navbar, then View License
2. Find the Release Channel line and click Change
   a. If the Change option is not available, sync your license; if this still doesn’t result in Change appearing, contact Plotly support
3. Select the channel you want to use
4. Click Save at the bottom of the screen
5. Click Dashboard in the navbar and click Check Now for updates
6. When the Server Manager reports there is an update, click View Update and Install Update
19. Admin panel reference

Dash Enterprise has a new administration panel as of version 3.4. You can access it in three possible ways after logging in with an account that has administrator privileges:

- Directly browse to https://plotly.your-domain.com/Auth/admin
- In the drop-down menu in the Dash Enterprise portal navbar, click App Manager, then click User in the new page’s navbar
- In the drop-down menu in the Chart Studio Enterprise navbar, click Admin Panel

19.1 Users overview

The Overview section of the Users tab features a summary of Dash users and license allocation and allows you to perform administrative tasks on user accounts.

Dash licenses is the maximum number of Dash App Developers (user accounts authorized to deploy Dash apps) for your license, and Dash licenses allocated to users is the number of Dash licenses being used. A user becomes a Dash App Developer by initializing a Dash app in the Dash Enterprise web portal. If all Dash licenses are in use, no new users will be able to create Dash apps, but existing Dash App Developers will continue to have app creation privileges.

Chart Creator licenses is the maximum number of chart creators (user accounts authorized to create charts in Chart Studio Enterprise) for your license, and Chart Creator licenses allocated to users is the number of users who have created charts. A user becomes a Chart Creator automatically upon saving a chart, unless all Chart Creator licenses are in use. You may free up a license by deleting all of an existing user’s charts.

A locked user account cannot save charts or Dash apps.
19.2 User summary table

<table>
<thead>
<tr>
<th>Username</th>
<th>User Created</th>
<th>Status</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>.-aA103</td>
<td>about 1 month ago</td>
<td>Active</td>
<td>Chart Creator</td>
</tr>
<tr>
<td><a href="mailto:Test_User_103@example.com">Test_User_103@example.com</a></td>
<td>about 18 hours ago</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.-aA104</td>
<td>about 1 month ago</td>
<td>Admin Active</td>
<td>Locked Apps: 6</td>
</tr>
<tr>
<td><a href="mailto:Test_User_104@example.com">Test_User_104@example.com</a></td>
<td>Never</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.-aA106</td>
<td>about 1 month ago</td>
<td>Active Locked</td>
<td>Chart Creator</td>
</tr>
<tr>
<td><a href="mailto:Test_User_106@example.com">Test_User_106@example.com</a></td>
<td>Never</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The User summary table lists all users in alphabetical order by default, but can also be sorted by date of user creation and time of user’s last login. The search function supports searches by username or email address. To access a user account’s settings, click on its row.

19.3 User settings panel

An administrator can perform the following actions on a user account in the User settings panel:

- Modify the email address
- Grant Administrator status
- Lock or unlock the account
- Grant or revoke a Chart Creator license
- Transfer the account’s files to another user
- Delete the account

To toggle Chart Creator status, transfer files from one user account to a different one, or delete a user account, an administrator must first lock the account in question.
19.4 Tasks tab

When you transfer/delete files or delete a user account, this job is entered into a queue. You can view the status of all jobs in the Tasks tab (https://plotly.your-company.com/Auth/admin/tasks). The queue will look similarly to below:
<table>
<thead>
<tr>
<th>User</th>
<th>Task Details</th>
<th>Status</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>_.aA104</td>
<td>Remove Chart Creator Privileges</td>
<td>Complete</td>
<td>Start: 5 days ago&lt;br&gt;End: 5 days ago</td>
</tr>
<tr>
<td>_.aA103</td>
<td>Transfer files to _.aA104</td>
<td>Complete</td>
<td>Start: 12 days ago&lt;br&gt;End: 12 days ago</td>
</tr>
<tr>
<td>_.aA106</td>
<td>Transfer files to _.aA103</td>
<td>Complete</td>
<td>Start: 13 days ago&lt;br&gt;End: 13 days ago</td>
</tr>
<tr>
<td>_.aA103</td>
<td>Remove Chart Creator Privileges</td>
<td>Complete</td>
<td>Start: 16 days ago&lt;br&gt;End: 16 days ago</td>
</tr>
<tr>
<td>_.aA103</td>
<td>Remove Chart Creator Privileges</td>
<td>Complete</td>
<td>Start: 22 days ago&lt;br&gt;End: 22 days ago</td>
</tr>
<tr>
<td>_.aA103</td>
<td>Remove Chart Creator Privileges</td>
<td>Complete</td>
<td>Start: 22 days ago&lt;br&gt;End: 22 days ago</td>
</tr>
<tr>
<td>dimadima</td>
<td>Remove Chart Creator Privileges</td>
<td>Complete</td>
<td>Start: about 1 month ago&lt;br&gt;End: about 1 month ago</td>
</tr>
</tbody>
</table>
20. Configure Dash Enterprise to use common SAML IdPs

The guide in this section is intended for general reference and should not be considered a replacement for documentation and training materials provided by your identity provider (IdP) vendor. When in doubt about how to configure your IdP, please refer to your IdP vendor’s guidance.

Dash Enterprise uses the SAML 2.0 standard to manage service provider (SP) initiated single sign-on (SSO). This process goes as follows:

1. The user’s browser requests a protected page in Dash Enterprise
2. Dash Enterprise redirects the request to the IdP for authentication
3. The IdP sends an HTML form containing a request for authentication to the user’s browser
4. The user submits credentials OR the user’s browser submits cached credentials
5. Dash Enterprise adds a request for user attributes to the credential submission
6. Upon successful authentication, the IdP returns an HTML form to the user’s browser containing the authentication assertion and user attributes requested by Dash Enterprise
7. The user’s browser POSTs the form to Dash Enterprise
8. Dash Enterprise authorizes the login and redirects the browser to the protected page

At this time, the IdPs types our customers have reported successfully integrating with Dash Enterprise are Active Directory Federation Services, PingFederate, and Okta. However, Dash Enterprise can integrate with any IdP supporting SAML 2.0 for authentication, so you are not limited to these options.

Before proceeding with configuring SAML:

- Ensure your Dash Enterprise server and IdP can make and receive network requests between each other
- Ensure your Dash Enterprise instance is using a certificate assigned by a certificate authority
  - If Plotly is hosting your instance at a *.plotly.host subdomain, it will already have an appropriate certificate

20.1 Active Directory Federation Services (AD FS)

1. On the server you intend to use as an IdP:
   a. Install the AD FS role service (guide) and enable the Web Server (IIS) role at the same time
   b. In IIS Manager, add an HTTPS binding to your Default Web Site:
      - Go to Server Certificates and create a self-signed certificate, then export it
Right-click **Default Website > Edit Bindings**, then add a new HTTPS binding using the above certificate

c. Configure the federation server (guide)

2. Browse to your Dash Enterprise Server Manager Settings and enable SAML:
   a. Select the following options:
      ■ Use SAML for authentication
      ■ No IdP metadata is available yet
      ■ Disable TLS/SSL certificate validation when communicating with the IdP
   b. Leave the remaining options as their default values
   c. Save the settings and restart when prompted
   d. When the app reports ready, return to Settings and copy the **SP (local) Metadata URL**
      ■ Ensure that your IdP can browse to this URL before proceeding

3. Return to your Active Directory server and:
   a. Create a claims-aware relying party trust (guide) using data imported from the **SP (local) Metadata URL** from the Dash Enterprise Settings
   b. Add a claims issuance policy with a rule sending the following LDAP attributes as claims (guide):
      ■ SAM-Account-Name attribute mapped to Name ID outgoing claim type
      ■ E-Mail-Addresses attribute mapped to EMail Address outgoing claim type
   c. Retrieve the **Federation Metadata** endpoint
      ■ You can find this in the AD FS snap-in’s > **Service > Endpoints** view
      ■ Append this value to your IdP’s hostname to get your IdP metadata URL

4. Return to your Server Manager settings and:
   a. In the SAML settings section, select **Enter a URL to the IdP (remote) metadata**
   b. In the **IdP (remote) Metadata URL** field, enter the Federation Metadata URL endpoint you retrieved above
   c. Save the settings and restart when prompted

### 20.2 PingFederate

1. In your PingFederate admin console, configure the credentials store you want to use (guide)
   a. For initial setup/testing with Dash Enterprise, the **Simple Username Password Credential Validator** is sufficient

2. Browse to your Server Manager Settings on port 8800 and enable SAML:
   a. Select the following options:
      ■ Use SAML for authentication
      ■ No IdP metadata is available yet
      ■ Disable TLS/SSL certificate validation when communicating with the IdP
   b. Leave the remaining options as their default values
   c. Return to the Dashboard and click **Stop**, then **Start** when it becomes available
   d. When the app reports ready, return to Settings and copy the **SP (local) Metadata URL**
      ■ Ensure that you can browse to this URL before proceeding
3. In your PingFederate admin console, create a new **SP connection** [guide] with an **SP adapter instance** [guide] and set options as follows:
   a. SP connection adapter with the connection type of Browser SSO
   b. Import metadata from URL using the **SP (local) Metadata URL** from Step 3d
   c. Browser SSO enabled with:
      - SP-initiated SSO and SP-initiated SLO
   d. Browser SSO > Assertion Creation with:
      - Standard identity mapping
      - No entries under Extend the Contract
   e. A new IdP Adapter Instance of type HTML Form Adapter that uses:
      - The credentials validator you set in Step 2
      - IdP adapter mapping: adapter source and username value
   f. An authentication policy contract using Authentication Policy Contract as a source and subject as a value
   g. Browser SSO > Protocol Settings > Allowable SAML Bindings using only POST and Redirect
   h. SP Connection > Credentials with a new certificate for signature verification

4. Click your new SP connection URL and set it to **Active**

5. Return to your Server Manager Settings and:
   a. In the SAML settings section, select Enter a URL to the IdP (remote) metadata
   b. In the **IdP (remote) Metadata URL** field, add your PingFederate IdP’s URL (see [guide] for how to identify it)
   c. Uncheck **Enable encryption**
   d. Check **Disable SSL certificate validation**
   e. Save the settings, then return to the Dashboard and click Stop, then Start

### 22.3 Okta

1. Browse to your Server Manager Settings on port 8800 and enable SAML:
   a. Select the following options:
      - Use SAML for authentication
      - No IdP metadata is available yet
      - Disable TLS/SSL certificate validation when communicating with the IdP
   b. Leave the remaining options as their default values
   c. Return to the Dashboard and click **Stop**, then **Start** when it becomes available
   d. When the app reports ready, return to Settings and copy the **SP (local) Metadata URL**
      - Ensure that you can browse to this URL before proceeding

2. Log into your Okta admin portal and configure SAML [guide] with the following changes from the default settings:
   a. Single sign on URL (or ACS URL): https://dash-your-domain.com/Auth/saml2/acs/
   b. Audience URI (SP Entity ID): the **SP (local) Metadata URL** from Step 3d
c. Name ID format: Unspecified

d. Application username: Email prefix

e. Click Next, then on Okta support, answer as follows and complete the wizard:
   - I’m an Okta customer adding an internal app
   - This is an internal app that we have created

3. Retrieve the Identity Provider metadata of the app you just created
   a. Click the Applications menu and the name of the app
   b. Click the Sign On tab and, on that page, right-click on Identity Provider metadata URL
      and copy the URL

4. Return to your Server Manager Settings and:
   a. In the SAML settings section, select Enter a URL to the IdP (remote) metadata
   b. In the IdP (remote) Metadata URL field, paste the Identity Provider Metadata from
      Step 3b
   c. Check Enable encryption when communicating with the IdP
   d. Save the settings, then return to the Dashboard and click Stop, then Start