LACEWORK

# Up and Running with Lacework and Jenkins

Lacework Editorial

August 18, 2020

In a recent blog post, I talked about how security teams need to focus on collaborating with Developers, Operations, and SREs by bridging the gap with relevant security data to drive testing and automation in CI/CD tooling. In this hands-on blog post, we will put that philosophy into practice by integrating Lacework Container Vulnerability Scanning APIs into a Jenkins pipeline to shift-left and scan images at build time.

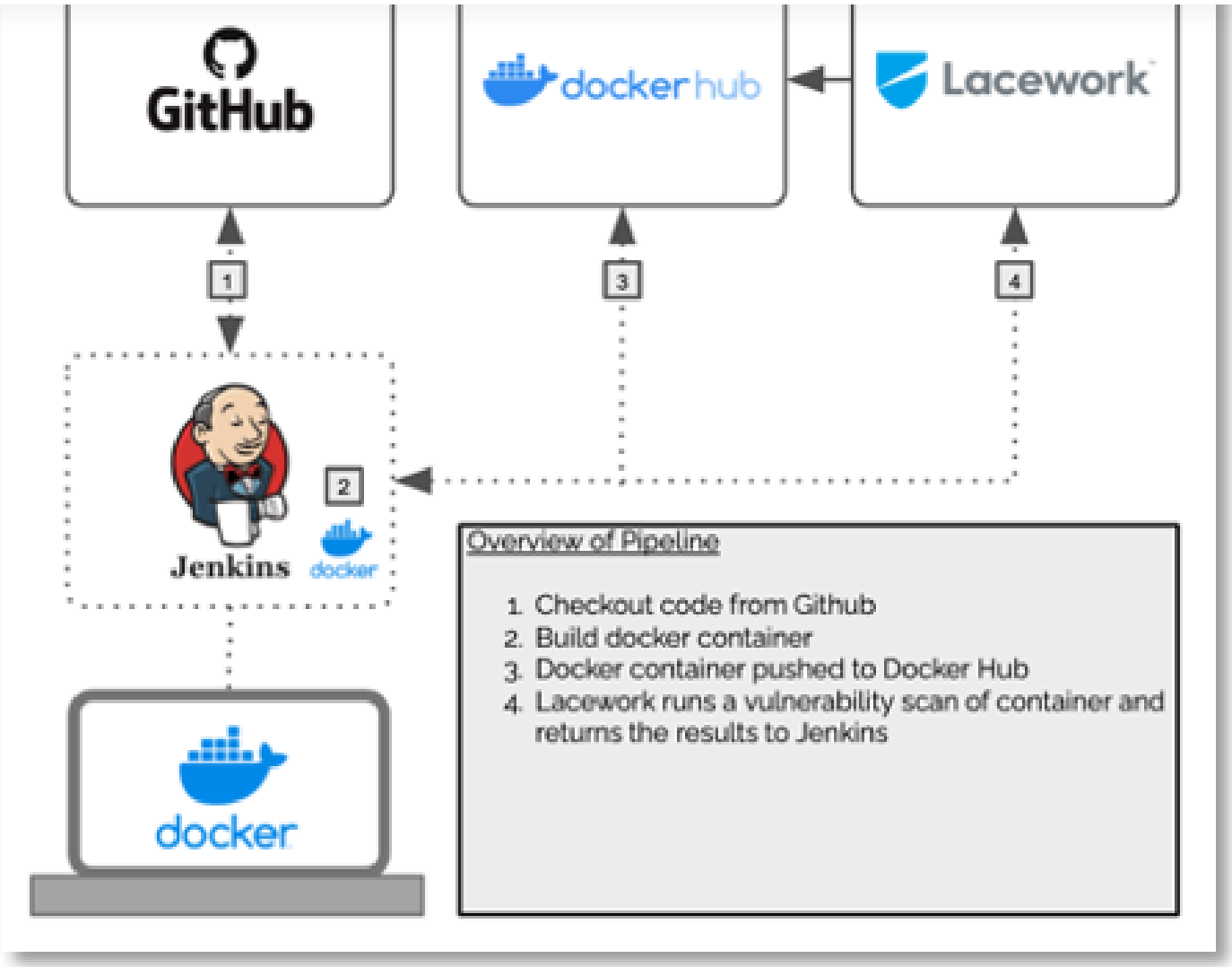
Jenkins is about as ubiquitous a product in DevOps tooling as there is. It is extremely versatile to configure continuous delivery pipelines to build and deploy just about any artifact out there, and it provides a ton of flexibility for integrating with other products. Additionally, it is also a great opportunity to "shift-left" and inject security earlier into the software development lifecycle (SDLC) through the use of automated testing, with the outcome being you spend less time, effort, and potentially significantly less money than trying to fix in production.

There are already a multitude of articles, books, and online tutorials on Jenkins, CI/CD, and Docker, so rather than try to cover each of those topics in any depth, this article is going to focus on a very simple pipeline design to understand the basics of integrating Jenkins and Lacework.

## What we are Going to Build

Before we begin, it is important to understand the high level workflow here. We are going to spin up a Jenkins pipeline that connects to GitHub and builds a Docker image from a Dockerfile, publishes the image to Docker Hub, and then initiates a container vulnerability scan of that image using the Lacework command line interface image.

LACEWORK




*We'll use Docker to provision Jenkins locally and connect to GitHub, Docker Hub, and Lacework!*

## No access to Jenkins?

NOT TO WORRY...We've got you covered there as well! We have created a Git repo that you will clone in a minute that contains a docker-compose file to provision Jenkins using Docker locally, and after applying a few configurations you will be good to go!

## Prerequisites

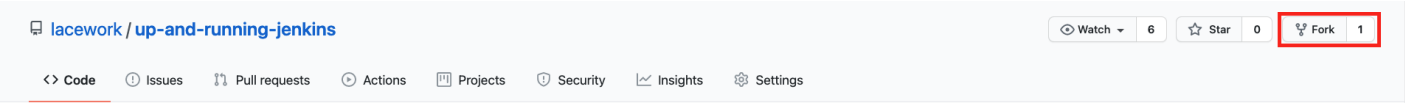
To run this tutorial you should feel comfortable on the command-line and have some very basic understanding of both Git, GitHub, and Docker. Additionally, there are a few items you'll need to have installed, and at the ready:

- A Lacework Account
- Git
- GitHub Account
- Docker Desktop
- Docker Hub Account

Once you have the prereqs checked off, you are good to go and we can get started!

LACEWORK

The first thing you will want to do is to fork the up-and-running-with-jenkins reference repo, and then clone it to your workstation with Git and Docker Desktop installed.



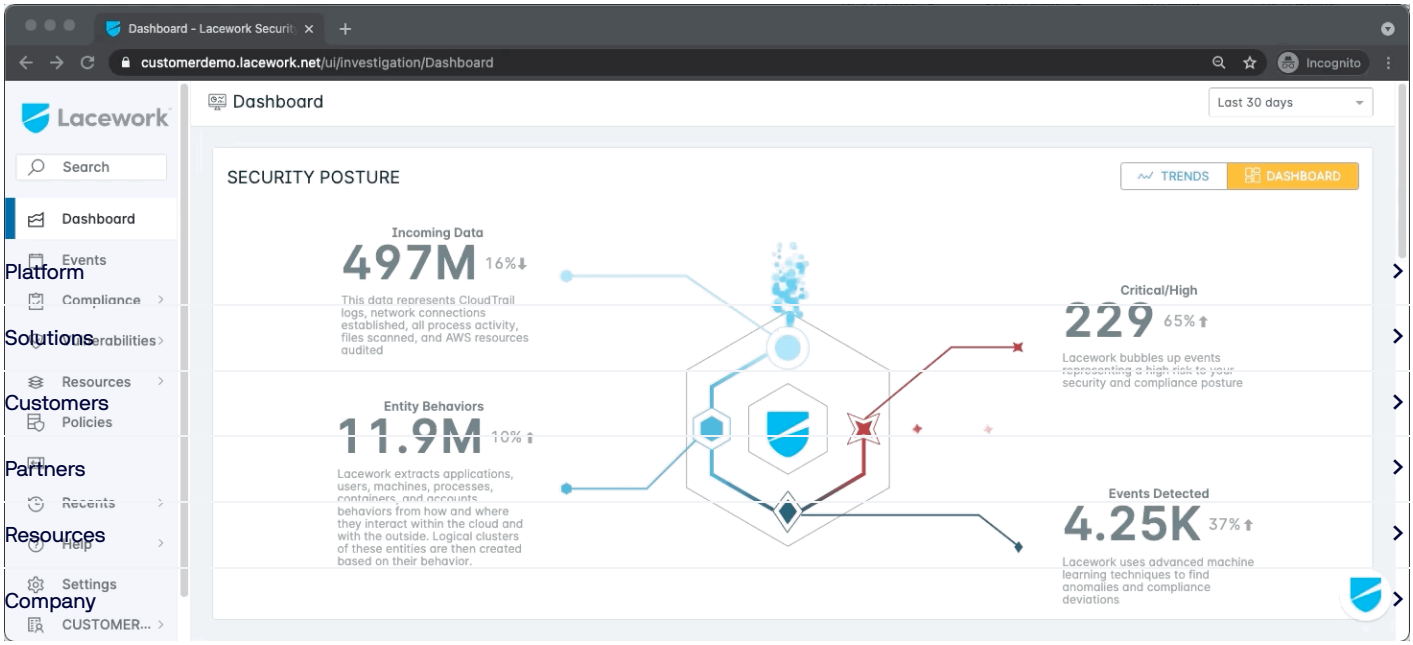

*You will need to fork the example repo to your GitHub account in order to connect Jenkins to it*

Open a terminal, clone the forked repo, and change directory into `up-and-running-jenkins` :

```
$ git clone https://github.com/<your_gh_username>/up-and-running-jenkins.git
$ cd up-and-running-jenkins
```

## Create Lacework API Key

To authenticate with Lacework and request on demand container vulnerability scans, we will need to hav




Request Free Trial

1. Log in to the Lacework Console.
2. Click **Settings -> API Keys** .
3. Click **CREATE NEW API KEY.**
4. Give the API key a **Name** and optional **Description.**
5. Click **SAVE.**

Search
Login
English

LACEWORK

The contents of your API key contain a `keyId`, `secret`, `subAccount`, and `account`:

```
{
  "keyId": "ACCOUNT_ABCEF01234559B9B07114E834D8570F567C824039756E03",
  "secret": "_abc1234e243a645bcf173ef55b837c19",
  "subAccount": "my-sub-account",
  "account": "my-account.lacework.net"
}
```

# Jenkins with docker-compose

The root of the jenkins-lacework-tutorial repo contains a `docker-compose.yml` file that we can use t

# docker-compose up

Platform

Make sure you are in the root directory of the project we cloned above and execute the following com

Solutions

Customers

```
$ docker-compose -p lacework up -d
Creating network "lacework_jenkins" with the default driver
Creating volume "lacework_jenkins-docker-certs" with default driver
Creating volume "lacework_jenkins-data" with default driver
Creating lacework_jenkins_1     ... done
Creating lacework_docker-dind_1 ... done
```

Partners

Resources

Company

Request Free Trial

Search

When Jenkins starts up the first time, it automatically creates an Administrator password that we wi

Login

English

```
$ docker exec lacework_jenkins_1 cat /var/jenkins_home/secrets/initialAdminPassword

```

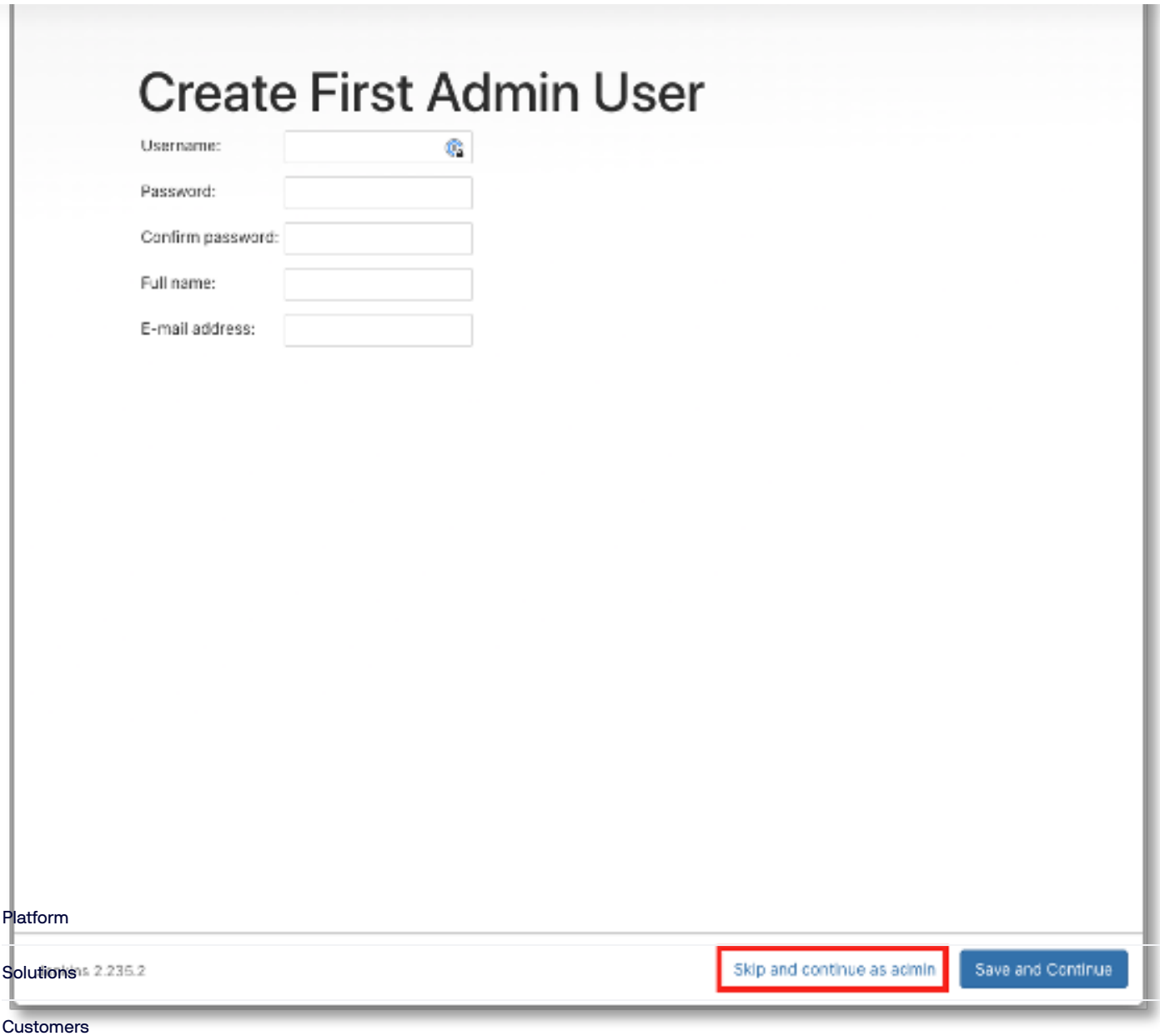
LACEWORK

Copy that password, open a web browser and go to **http://localhost:8080**


Paste the Administrator password and click **Continue**

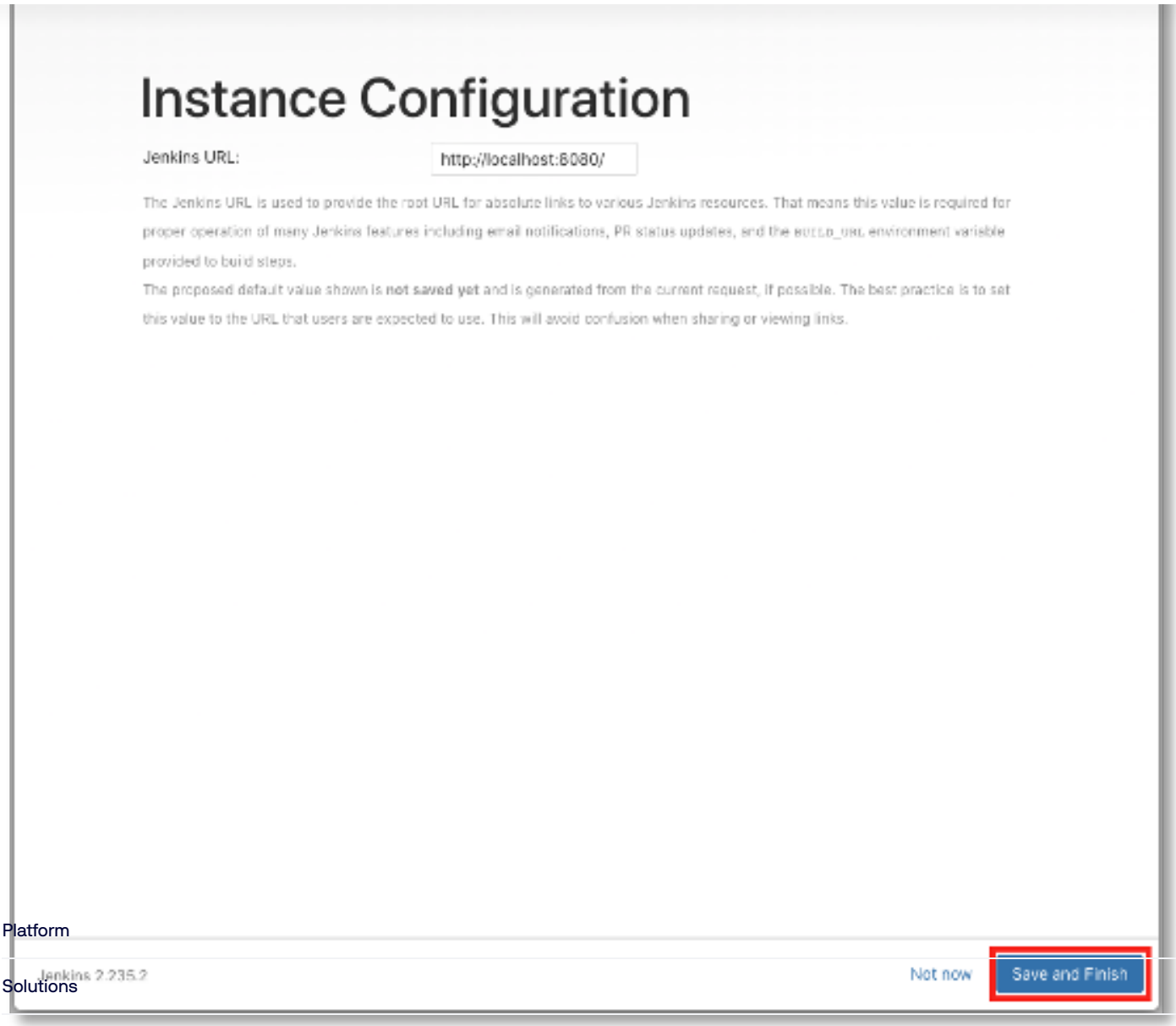
Next, click **Install suggested plugins.** This process takes about a minute or so depending on your Int

LACEWORK



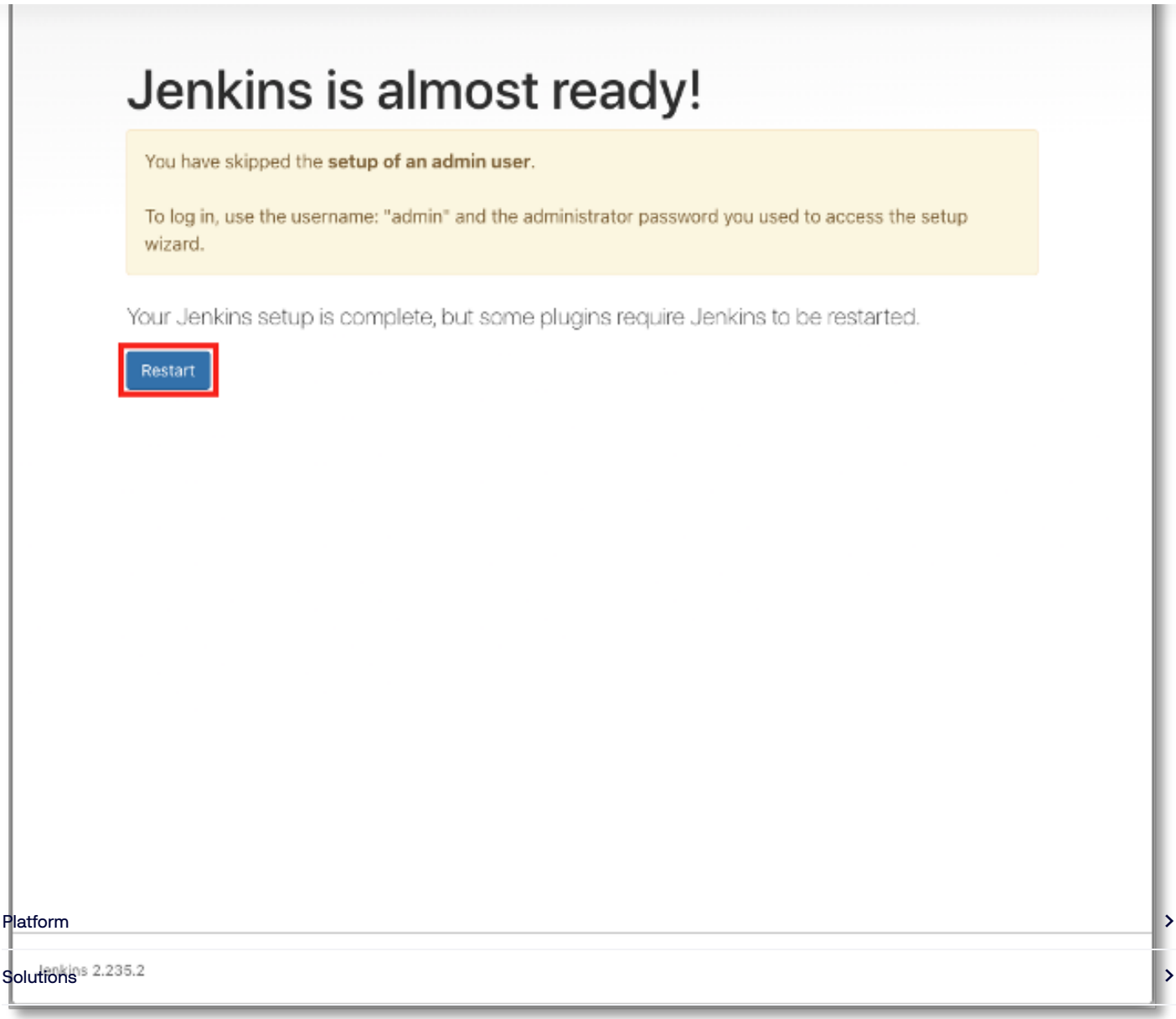

After the plugins are installed you have the option to create an Admin User, or you can just use the

LACEWORK




Next click **Save and Finish.**

LACEWORK




Platform

Solutions

Customers

Partners

Resources

Company

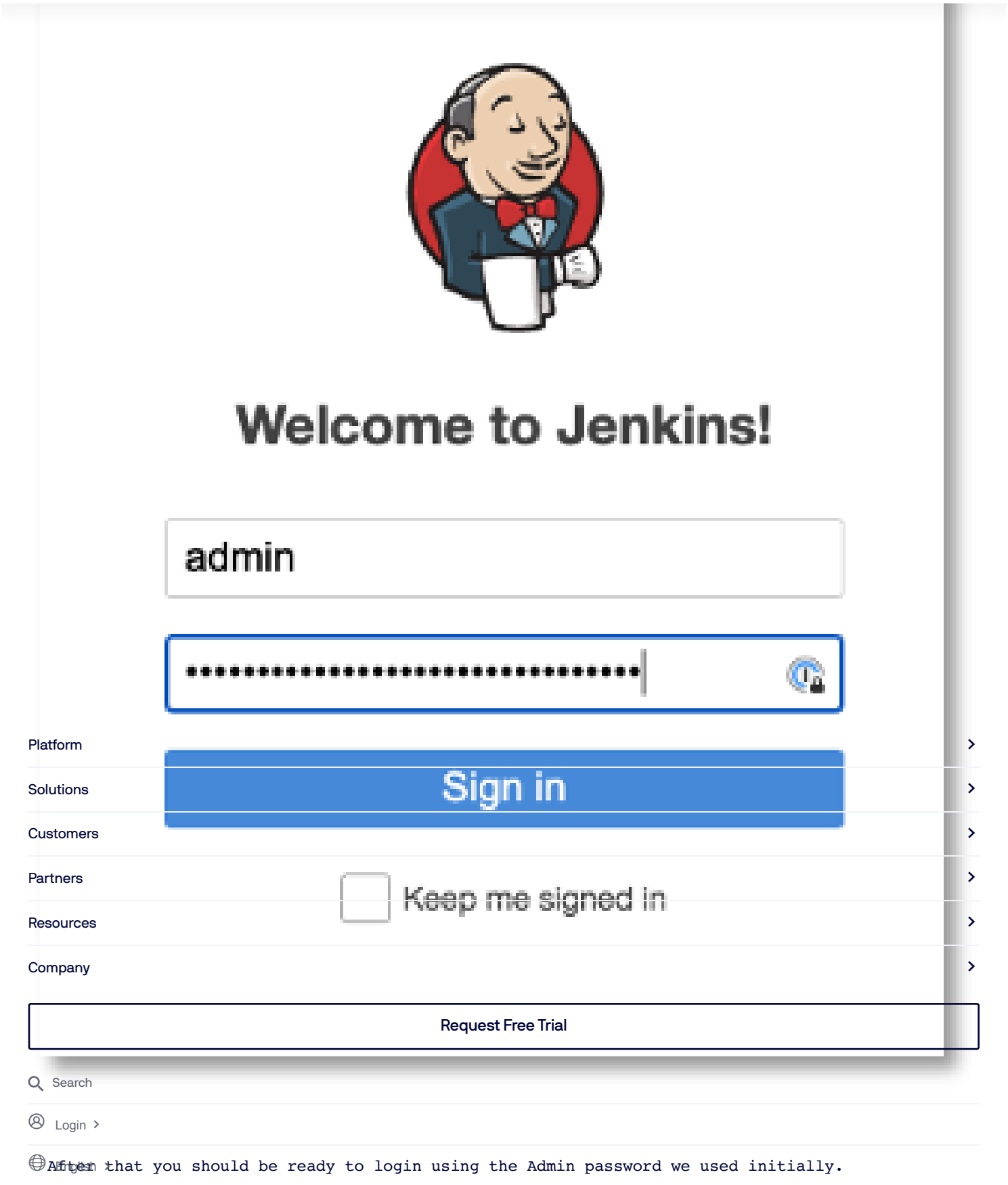
Request Free Trial

Search

Login

English

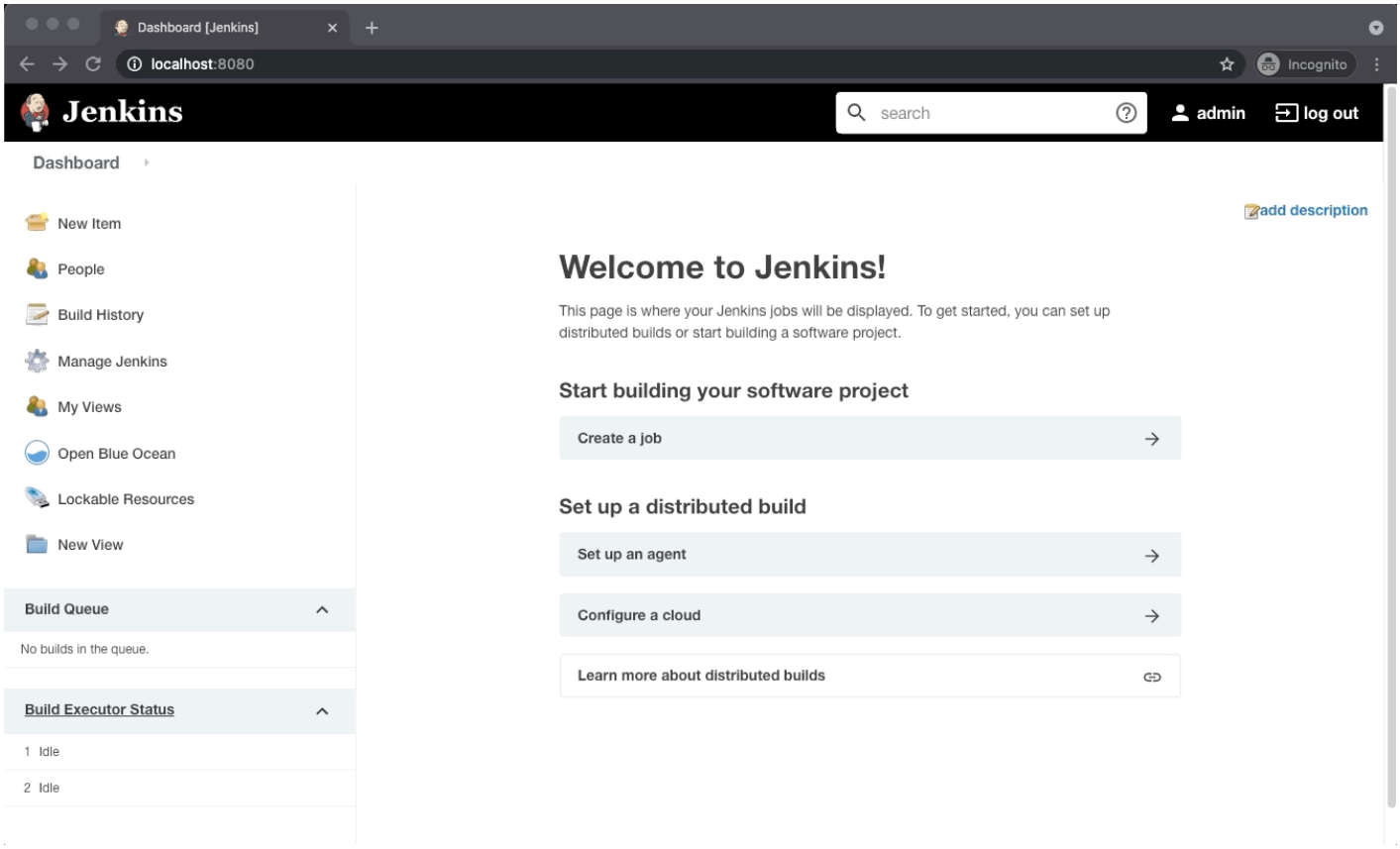
Now, just click **Restart** to finish the initial setup. (Note: you may need to Refresh your browser to




After that you should be ready to login using the Admin password we used initially.

## Install Docker Pipeline Plugin

LACEWORK




You will need to install **Docker Pipeline** plugin. This plugin allows you to build, test, and use Dock



## Configuring Jenkins

With Jenkins running locally we are ready to configure the pipeline. The repository that you cloned

## Environment Variables

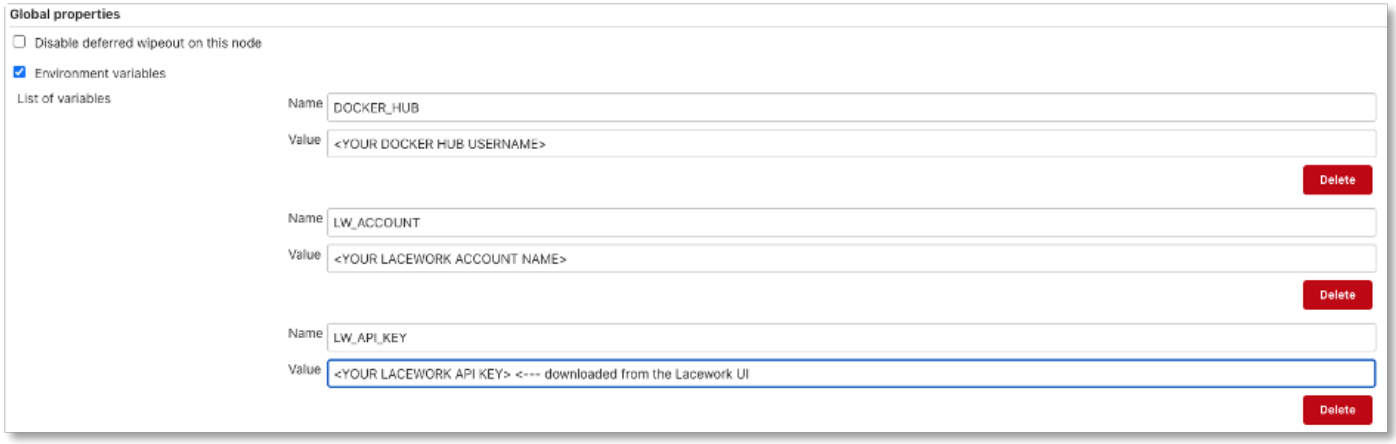
We are going to define three global environments variables. Those variables are as follows:

- LW_ACCOUNT — The name of your Lacework account.
- LW_API_KEY — API Key ID from the downloaded JSON file.

LACEWORK

**NOTE:** Environment variable names are case sensitive.

Open the **Manage Jenkins -> Configure System** page and then scroll down to **Environment variables** and c

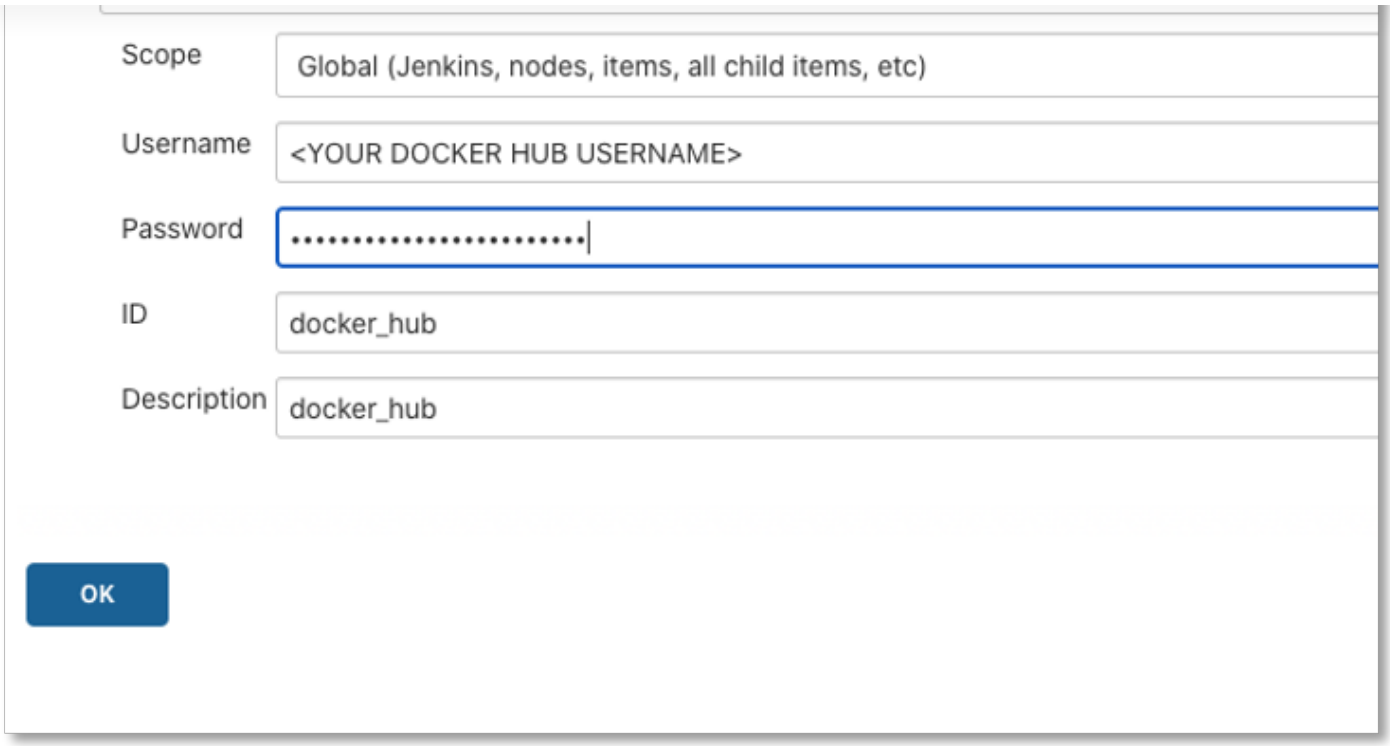



## Credentials

You will need to add your username and password for Docker Hub, as well as another type of credentia

Add your Docker Hub Credentials as follows and use the ID 'docker_hub' as that is how it is referred

LACEWORK

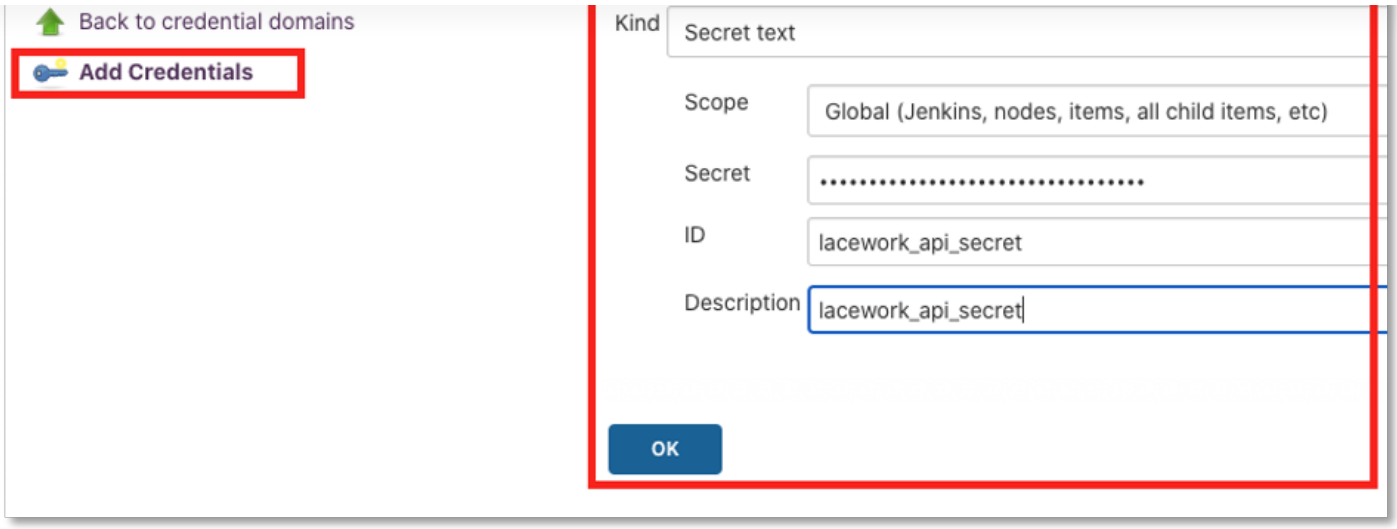



Click **Ok.**

Next, on the left click **Add Credentials** and add the following credentials:

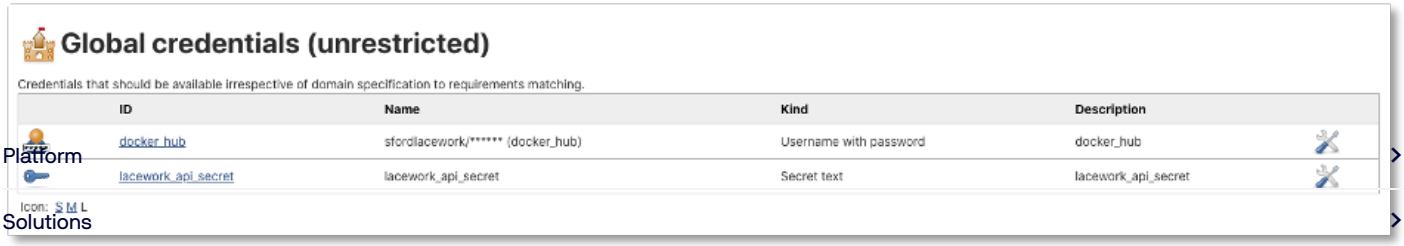
- **Kind:** Secret Text
- **Scope:** Global
- **Secret:** *<Paste your Lacework API Secret from the downloaded JSON>*
- **ID:** lacework_api_secret
- **Description:** lacework_api_secret

LACEWORK



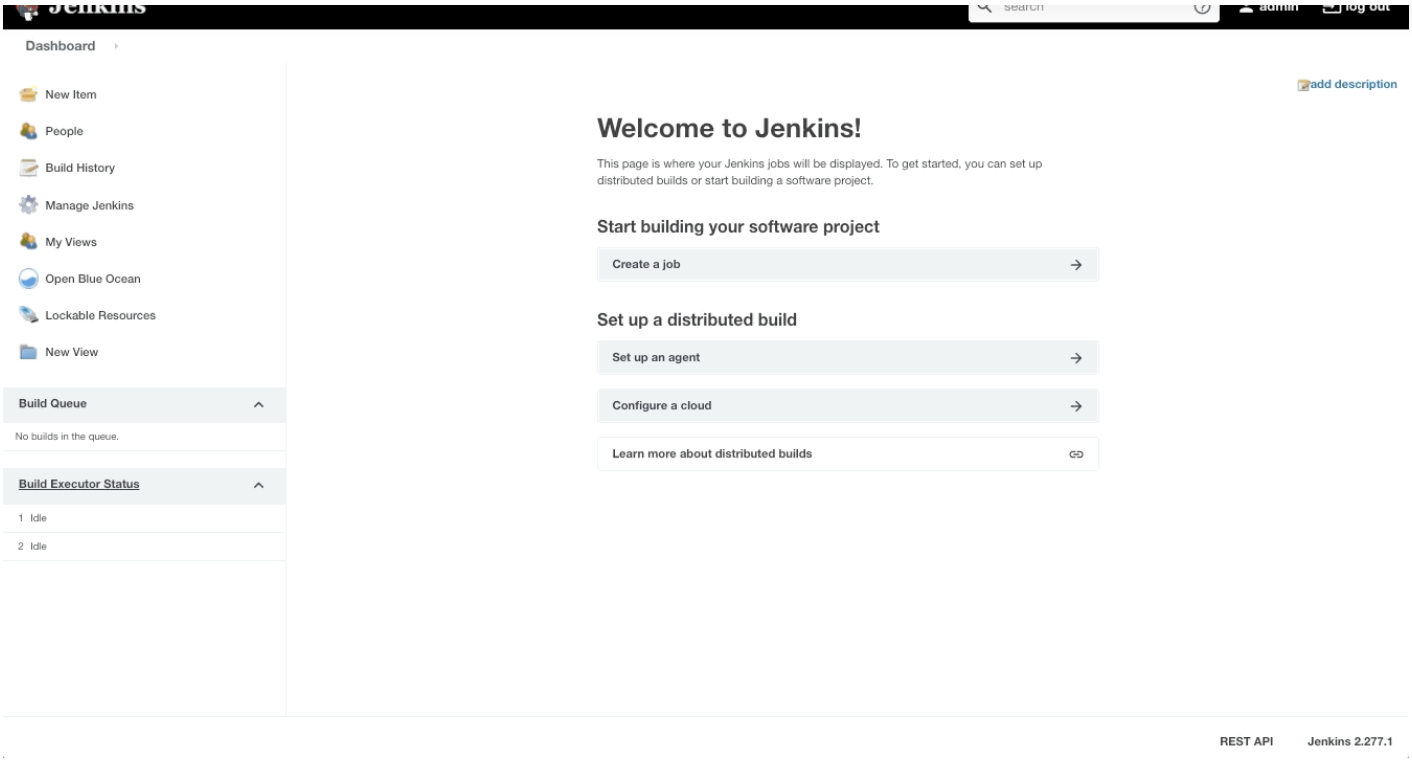

Click **Ok.**

At this point you should see the two credentials stored in Jenkins:




We are now ready to move on to configuring the pipeline.

## Create Pipeline with Blue Ocean




The version of Jenkins we used with our Docker container has a user interface called Blue Ocean inst

1. On the left hand side click **Open Blue Ocean.**
2. Next click **Create New Pipeline**
3. When asked where you store your code choose GitHub.
4. Next we need to allow Jenkins to monitor our repo for changes and for that we are going to need to
5. You will be redirected to login to GitHub and immediately taken to the Personal Access Token gener
6. Copy the token to your clipboard and paste it back in Jenkins.
7. Once authenticated you should be able to navigate any of the Github Orgs you have access to so you
8. Select the **up-and-running-jenkins** repo from your list of repositories, and then click **Create Pipel**
9. Jenkins will automatically find the Jenkinsfile in the root of the directory and kick off the pipe

LACEWORK


You can click on the pipeline to watch the build, publish and finally the scan from Lacework!

The output here shows a human readable summary of the vulnerabilities found. If no vulnerabilities a

# Spinning Jenkins Down



Once you are ready to tear down the test environment you have two options.

## Save the State

If you are going to continue to play around with Jenkins as it is configured and want to save the st

```
docker-compose -p lacework down
```



When you are ready to continue your work you just run:

LACEWORK

## Delete State

If you want to complete tear down Jenkins and remove the docker volumes and network, run the followi

```
docker-compose -p lacework down --volumes
```

## Conclusion

Injecting security into your SDLC should be a priority. Companies adopting these best practices are

Stay tuned for more episodes of Up and Running with Lacework. Until then, happy automating!

LACEWORK

Documentation

## Support

Support

Status

Login

## Contact Us

To request a demo or chat with the sales team:

Contact Us