



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.

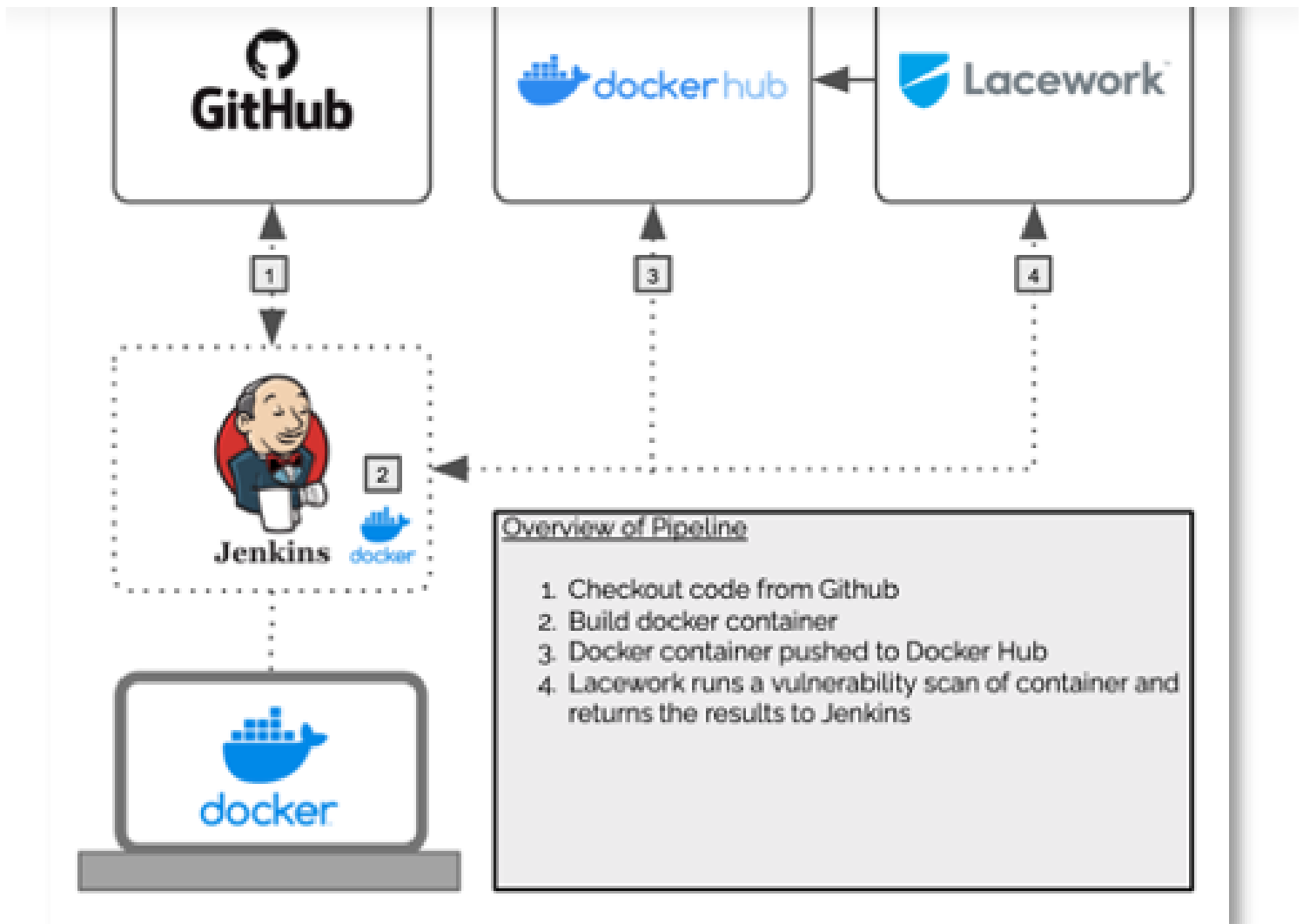
Platform	>
Solutions	>
Customers	>
Partners	>
Resources	>
Company	>

[Request Free Trial](#)

Search

Login >

English >



Platform

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

Solutions

No access to Jenkins?

Customers

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!

Resources

Company

Prerequisites

Request Free Trial

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:

Q Search

◦ [A Lacework Account](#)

⊗ Login > [Git](#)

◦ [GitHub Account](#)

🌐 English [Docker Desktop](#)

◦ [Docker Hub Account](#)

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



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.



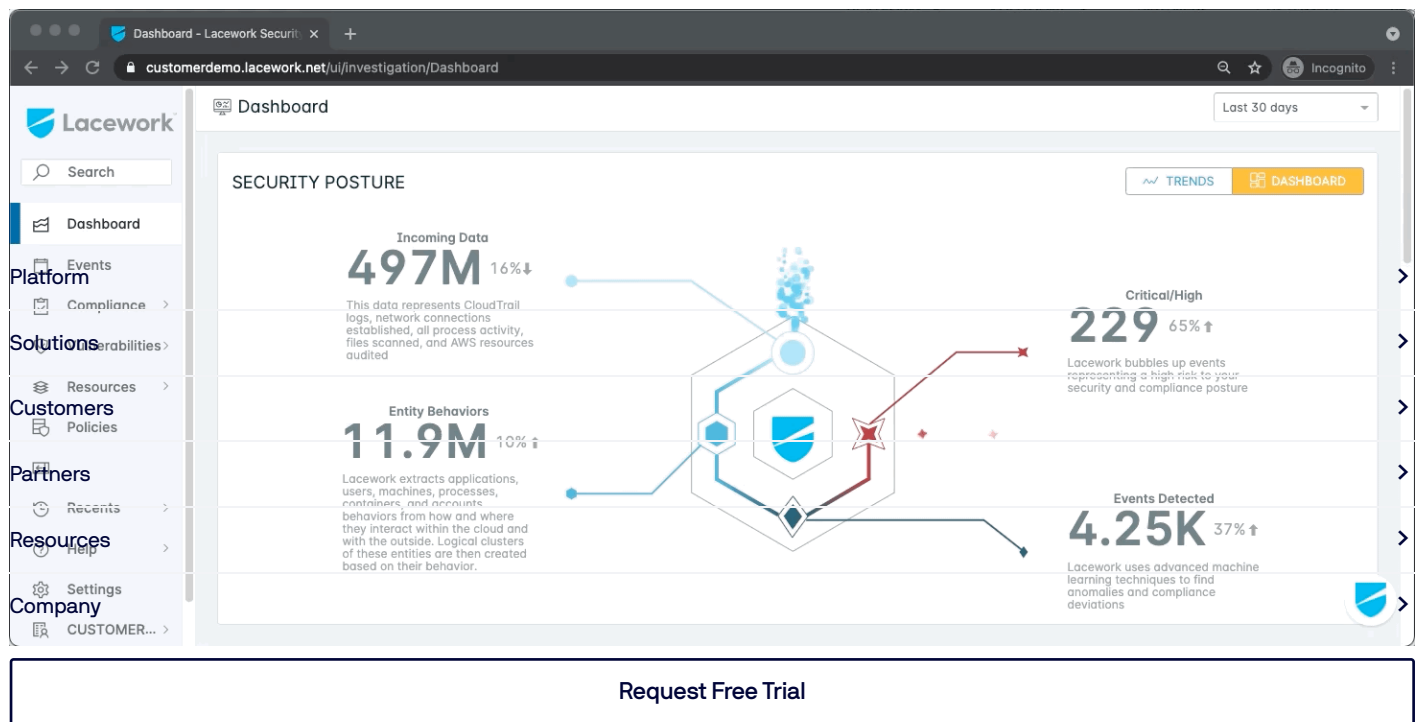
You will need to fork the example repo to your GitHub account in order to connect Jenkins to it

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

```
1 $ git clone https://github.com/<your_gh_username>/up-and-running-jenkins.git
2 $ 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



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**.



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

```
1 {
2   "keyId": "ACCOUNT_ABCEF01234559B9B07114E834D8570F567C824039756E03",
3   "secret": "_abc1234e243a645bcf173ef55b837c19",
4   "subAccount": "my-sub-account",
5   "account": "my-account.lacework.net"
6 }
```

Jenkins with docker-compose

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

docker-compose up

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

Solutions

Customers

Partners

Resources

Company

1 \$ docker-compose -p lacework up -d

2 Creating network "lacework_jenkins" with the default driver

3 Creating volume "lacework_jenkins-docker-certs" with default driver

4 Creating volume "lacework_jenkins_data" with default driver

5 Creating lacework_jenkins_1 ... done

6 Creating lacework docker-dind 1 ... done

Request Free Trial

Search

When Jenkins starts up the first time, it automatically creates an Administrator password that we will use to login

English

```
1 $ docker exec lacework_jenkins_1 cat /var/jenkins_home/secrets/initialAdminPassword
2
```



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

- Platform >
- Solutions >
- Customers >
- Partners >
- Resources >
- Company >

Request Free Trial

Search

Login >

English >



Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Platform

Solutions Jenkins 2.235.2

Customers

Partners

Resources

Company

[Skip and continue as admin](#)

[Save and Continue](#)

[Request Free Trial](#)

Search

Login >

English >



Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `JENKINS_URL` environment variable provided to build steps.

The proposed default value shown is not saved yet and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Platform

Solutions

Jenkins 2.235.2

Not now

Save and Finish

Customers

Partners

Resources

Next click **Save and Finish**.

Company

Request Free Trial

Search

Login >

English >



Jenkins is almost ready!

You have skipped the **setup of an admin user**.

To log in, use the username: "admin" and the administrator password you used to access the setup wizard.

Your Jenkins setup is complete, but some plugins require Jenkins to be restarted.

[Restart](#)[Platform](#)[Solutions](#)[Customers](#)[Partners](#)

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

[Resources](#)[Company](#)[Request Free Trial](#)[Search](#)[Login](#)[English](#)



Welcome to Jenkins!



Platform >

Solutions >

Customers >

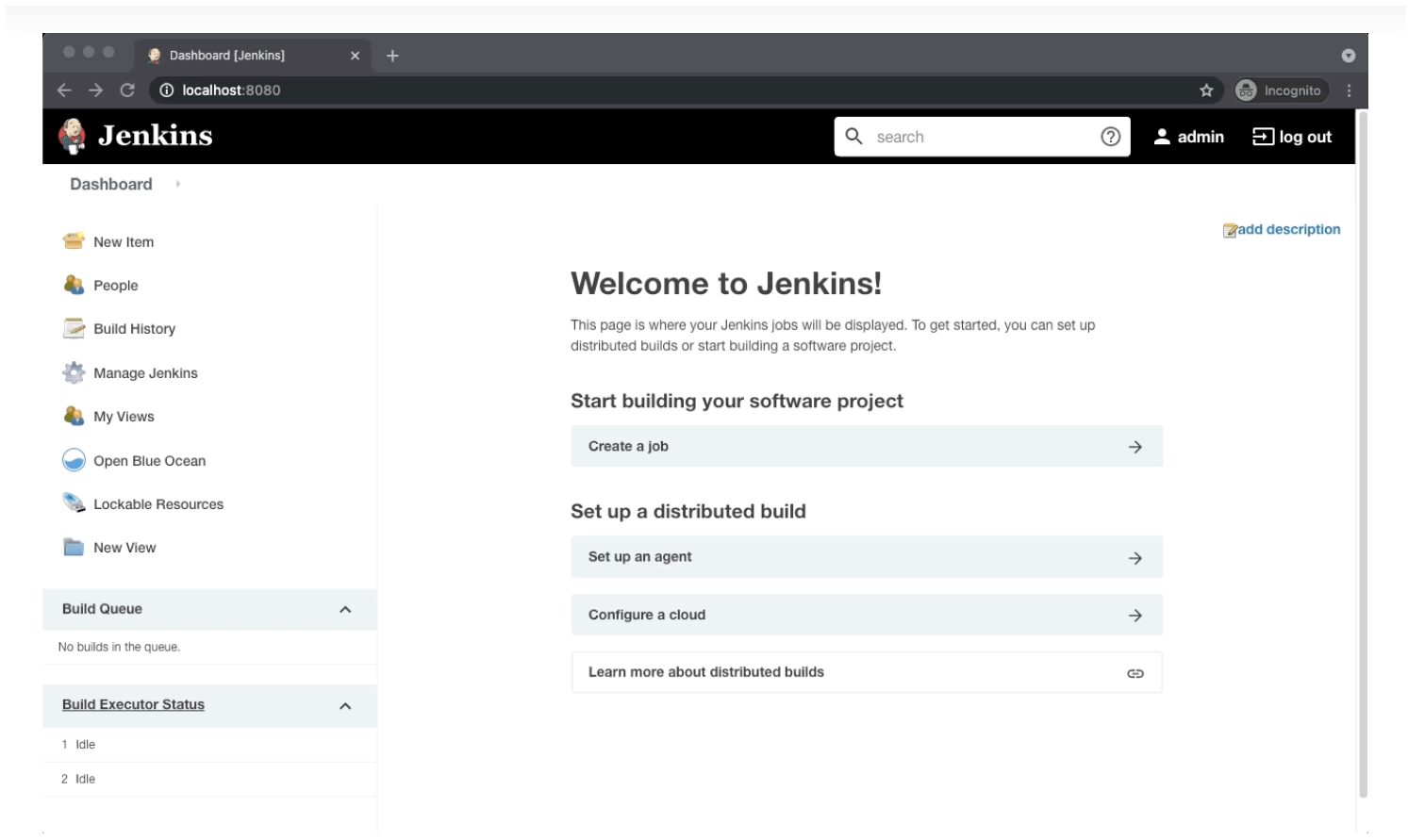
Partners >

Resources >

Company >

☐ Keep me signed in

Install Docker Pipeline Plugin



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

Platform >

Configuring Jenkins >

Solutions >

Customers >

Partners >

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

Resources >

Environment Variables >

Request Free Trial

Search

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

Login >

English >

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



NOTE: Environment variable names are case sensitive.

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

Global properties

☐ Disable deferred wipeout on this node

☒ Environment variables

List of variables

Name

DOCKER_HUB

Value

<YOUR DOCKER HUB USERNAME>

Delete

Name

LW_ACCOUNT

Value

<YOUR LACEWORK ACCOUNT NAME>

Delete

Name

LW_API_KEY

Value

<YOUR LACEWORK API KEY> <--- downloaded from the Lacework UI

Delete

Credentials

Now you will need to add your username and password for Docker Hub, as well as another type of credential

- Solutions

>
- Customers

>
- Partners

>
- Resources

>
- Company

>

Request Free Trial

- Search
- Login >
- English >



Scope Global (Jenkins, nodes, items, all child items, etc)

Username <YOUR DOCKER HUB USERNAME>

Password

ID docker_hub

Description docker_hub

OK

Click **Ok**.

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

Platform	>
• Kind: Secret Text	>
Solutions	>
• Scope: Global	>
Customers	>
• Secret: <Paste your Lacework API Secret from the downloaded JSON>	>
Partners	>
• ID: lacework_api_secret	>
Resources	>
• Description: lacework_api_secret	>
Company	>

Request Free Trial

Search

Login >

English >



Back to credential domains

Add Credentials

Kind: Secret text

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret:

ID: lacework_api_secret

Description: lacework_api_secret

OK

Click **Ok**.

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

Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
docker_hub	sfordlacework/***** (docker_hub)	Username with password	docker_hub
lacework_api_secret	lacework_api_secret	Secret text	lacework_api_secret

Icon: Solutions

Customers

Partners

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

Company

Create Pipeline with Blue Ocean

Request Free Trial

Search

Login

English



REST API Jenkins 2.277.1

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 Pipeline**.

9. Jenkins will automatically find the Jenkinsfile in the root of the directory and kick off the pipe

Q Search

Login >

English >



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

Platform >

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

Solutions >

Save the State >

Partners >

Resources >

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

Request Free Trial

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

Search

Login >

English >

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



Delete State

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


```
1 | 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!

Platform

Solutions

Customers

Explore

Partners

Platform

Resources

Solutions

Company

Threat Detection

Vulnerability Management

Container Security

Search

Multicloud

Cloud Security Posture and Compliance

Polygraph Experience

English >

Company

About Us

Investors

Events

Press Releases

Careers

Legal

Request Free Trial

Cookies Settings

Learn

Blog



Documentation

Support

Support

Status

Login

Contact Us

To request a demo or chat with the sales team:

Contact Us



© 2022, Lacework, All Rights Reserved. [Privacy Policy](#) | [Terms of Use](#)