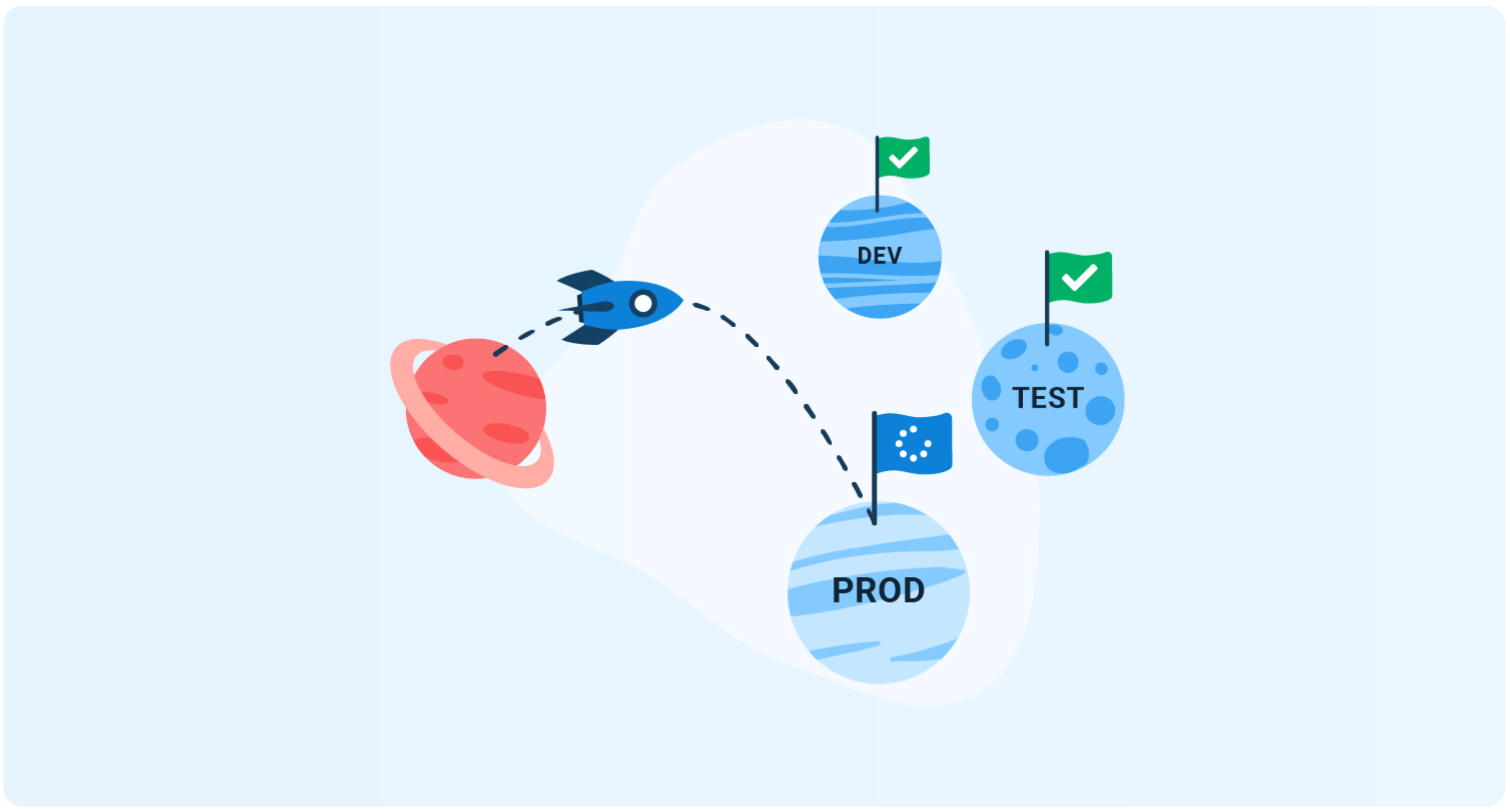


Multi-environment deployments with Jenkins and Octopus

Terence Wong
February 22, 2022 • 5 mins

During a deployment process, an artifact is built by a build server before being deployed. Jenkins is a build server designed for multi-environment settings. Jenkins can package and push your artifact to a central repository. From here, a Continuous Delivery (CD) tool can take the artifact and deploy it.

Octopus Deploy is a best in class CD tool that helps with this process. Octopus can interface with and deploy to major cloud providers like Azure, Google, and Amazon.

In this post, I show you how to build and push the Octopus underwater app to Amazon Elastic Container Registry (ECR). Jenkins will trigger a deployment in Octopus Deploy. Octopus will then deploy the app to Amazon Elastic Kubernetes Service (EKS).

Prerequisites

To follow along, you need:

- An Amazon Web Services (AWS) account
- A GitHub account
- A Jenkins instance set up with a pipeline. If you're using the branch, specify the jenkins-octopus branch reference in Jenkins.

This post uses the Octopus underwater app repository. You can fork the repository and use the main branch to follow along.

The jenkins-octopus branch contains the template files needed to complete the steps in this post. You have to replace some values with your own, but I've included my values in this post as a reference.

Amazon Web Services setup

To set up AWS for Jenkins, you need to create an access key and an ECR repository to store the image.

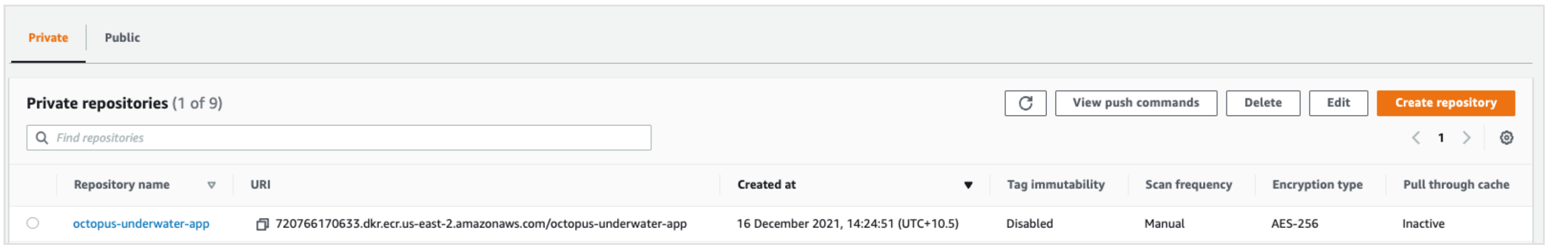
To create an access key, go to **Amazon Console**, then **IAM**, then **Users**, `[your user]`, then **Security credentials**, and then **Create Access Key**.

Your browser downloads a file containing the Access Key ID and the Secret Access Key. These values are used in Jenkins to authenticate to Amazon.

To create a repository, go to the **Amazon Console**, then **ECR**, and then **Create Repository**.

You need to set up an image repository for each image that you publish. Give the repository the same name you want the image to have.

You'll see your repository under **Amazon ECR**, then **Repositories**. Make a note of the zone it's in, in the URI field.



Private repositories (1 of 9)

	Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type	Pull through cache
	octopus-underwater-app	720766170633.dkr.ecr.us-east-2.amazonaws.com/octopus-underwater-app	16 December 2021, 14:24:51 (UTC+10.5)	Disabled	Manual	AES-256	Inactive

AWS cluster setup

Set up the cluster in AWS using the guide in our previous post, Creating an ESK cluster in AWS.

Extend the pipeline with Octopus `release` and `deploy` commands. Create a Jenkinsfile and paste the following code:

```
pipeline {
    agent any
    options {
        skipStagesAfterUnstable()
    }
    stages {
         stage('Clone repository') {
            steps {
                script{
                checkout scm
                }
            }
        }

        stage('Build') {
            steps {
                script{
                 app = docker.build("octopus-underwater-app")
                }
            }
        }
        stage('Test'){
            steps {
                 echo 'Empty'
            }
        }
        stage('Push') {
            steps {
                script{
                        docker.withRegistry('https://720766170633.dkr.ecr.us-east-2.amazonaws.co
                    app.push("${env.BUILD_NUMBER}")
                    app.push("latest")
                    }
                }
            }
        }
        stage('Deploy'){
            steps {
                script{
                    octopusCreateRelease additionalArgs: '', cancelOnTimeout: false, channel: ''
                    octopusDeployRelease cancelOnTimeout: false, deploymentTimeout: '', environm
                }
            }
        }

    }
}
```

Octopus Deploy setup

In your Octopus Deploy instance, you need to create a project called `aws-jenkins`:

- Go to **Project**, then **Add Project**.
- Add the `aws-jenkins` title and click **Save**.

Next, set up Development, Test, and Production environments:

- Go to **Infrastructure**, then **Environments**, and then **Add Environment**.

- Name it `Development` and click **Save**.
- Do the same for a Test and Production environment.

You need to set up the Amazon account to deploy to EKS:

- Go to **Infrastructure**, then **Accounts**, then **Add Account**, and then **AWS Account**.
- Give it a name and fill out the **Access Key ID** and **Secret Access Key** from earlier.

Now you need to set up your AWS Kubernetes cluster as a deployment target in Octopus Deploy:

- Go to **Infrastructure**, then **Deployment Targets**, then **Add Deployment Target**, then **Kubernetes Cluster**, and then **Add**.

Follow the steps in our Docs which indicate the fields to add to set up the deployment target. In this section you give the deployment target a target role. This will be referenced in the Octopus Deploy step later.

You need to add the Amazon feed to the Octopus instance:

- Go to **Library**, then **External Feeds**, then **Add Feed**, and then select the **AWS Elastic Container Registry**.
- Enter your **Access Key ID**, **Secret Access Key**, and the **Zone** of your registry.

Deploying to EKS step

In your `aws-jenkins` project, go to **Process**, then **Add deployment step**, then **Kubernetes**, and then **Deploy Kubernetes Containers**. Add the target role that you gave your deployment target earlier.

Add the following into the YAML section:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: octopus-underwater-app-jenkins
  labels:
    app: octopus-underwater-app
spec:
  selector:
    matchLabels:
        app: octopus-underwater-app
  replicas: 3
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: octopus-underwater-app
    spec:
      containers:
        - name: octopus-underwater-app
          image: 720766170633.dkr.ecr.us-east-2.amazonaws.com/octopus-underwater-app
          ports:
            - containerPort: 80
              protocol: TCP
          imagePullPolicy: Always
```

Click **SAVE**.

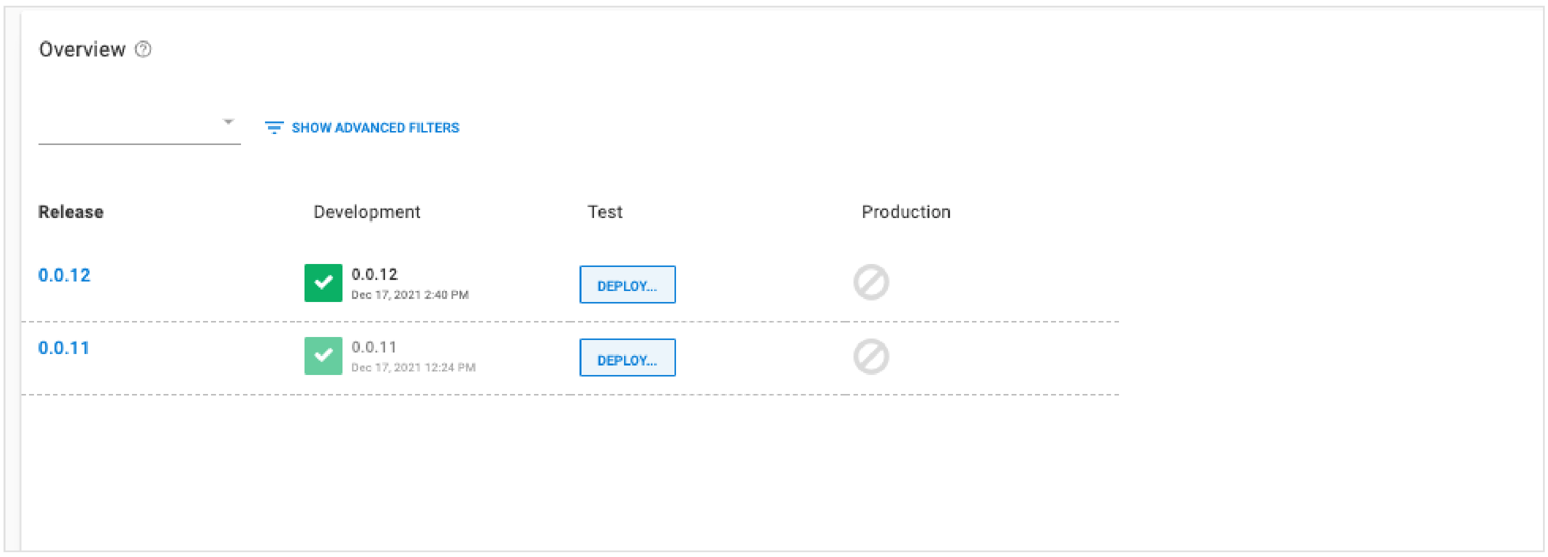
Deploying in Jenkins

To connect Jenkins to Octopus, you need to install the Octopus Deploy plugin.

Go to **Dashboard**, then **Manage Plugins**. If the plugin is not installed, search for the Octopus plugin in the **Available** tab and install the plugin. Follow the guide in our Docs that explains how to set up Jenkins and the Octopus plugin.

After the Octopus plugin is set up, making a change to the code in GitHub will trigger a build in Jenkins and Octopus.

Navigate back to your Octopus instance and **Project**, then **Overview**, and you'll see the release deployed to the Development environment.

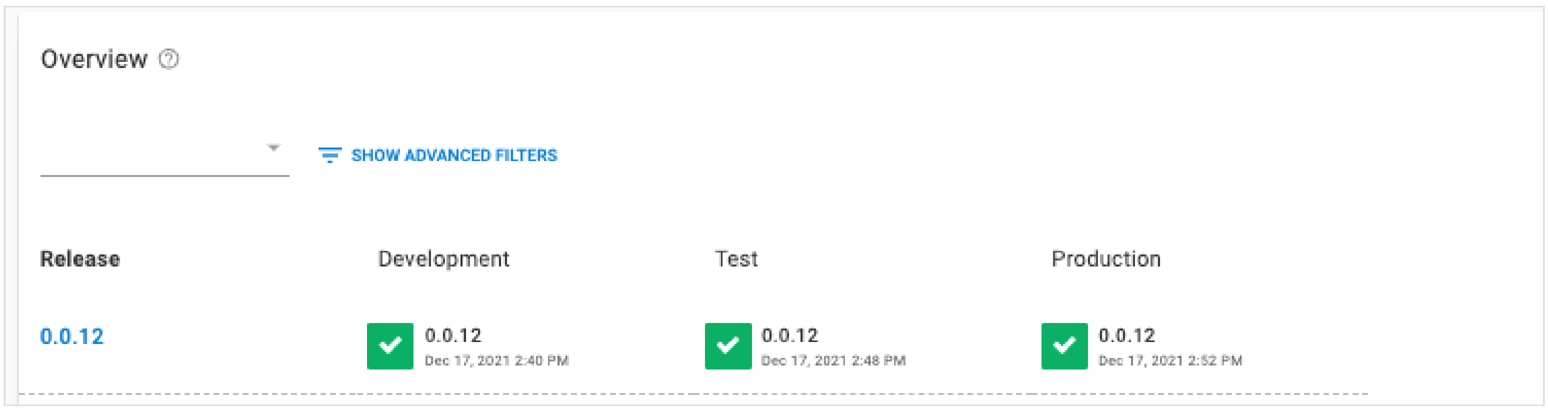


Overview

SHOW ADVANCED FILTERS

Release	Development	Test	Production
0.0.12	✓ 0.0.12 Dec 17, 2021 2:40 PM	DEPLOY...	⊘
0.0.11	✓ 0.0.11 Dec 17, 2021 12:24 PM	DEPLOY...	⊘

Now you can progress the release to the Test and Production environment when you're ready. Click **Deploy** to progress the release.



Overview

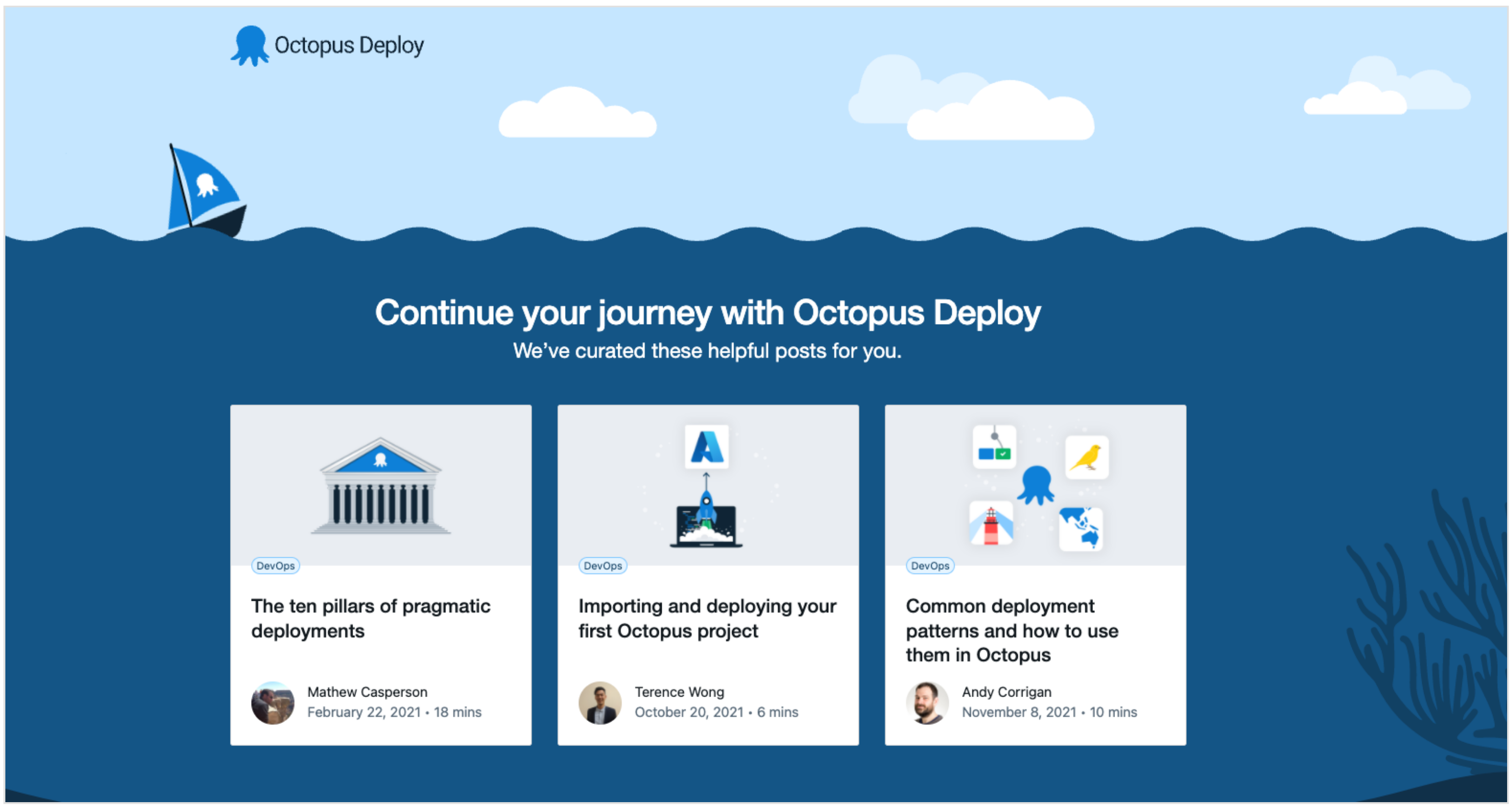
SHOW ADVANCED FILTERS

Release	Development	Test	Production
0.0.12	✓ 0.0.12 Dec 17, 2021 2:40 PM	✓ 0.0.12 Dec 17, 2021 2:48 PM	✓ 0.0.12 Dec 17, 2021 2:52 PM

You need to port forward locally to inspect the service. Use this command to inspect the web application. The port, 28015, is based on the example in the Kubernetes documentation:

```
kubectl port-forward deployment/octopus-underwater-app-jenkins  28015:80
```

Go to the IP address `http://127.0.0.1:28021/` in your browser to view your web application.

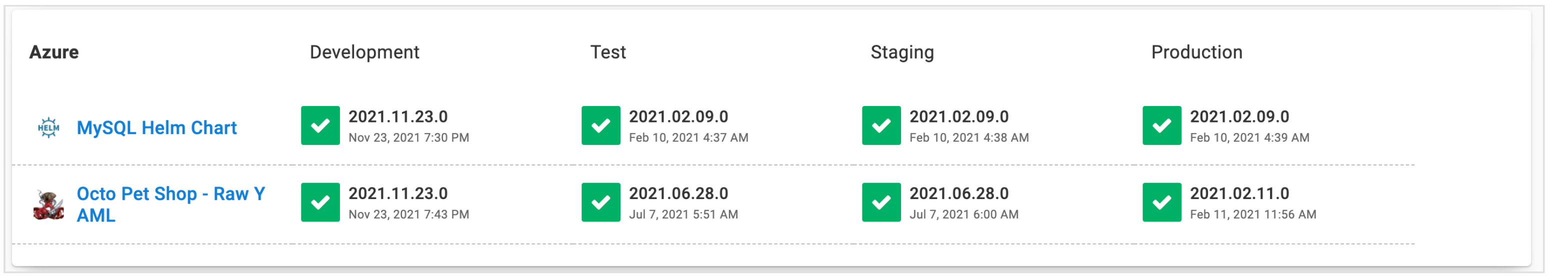

Octopus Deploy
Continue your journey with Octopus Deploy
We've curated these helpful posts for you.
DevOps
The ten pillars of pragmatic deployments
Mathew Casperson
February 22, 2021 • 18 mins
DevOps
Importing and deploying your first Octopus project
Terence Wong
October 20, 2021 • 6 mins
DevOps
Common deployment patterns and how to use them in Octopus
Andy Corrigan
November 8, 2021 • 10 mins

The benefits of a dedicated CD tool

Octopus Deploy is a dedicated Continuous Delivery (CD) tool that natively supports release management. Jenkins defines environments through the pipeline file. They are dividers to the pipeline code.

In Octopus, environments are dedicated spaces. Octopus Deploy makes it easy to stop a deployment in a staging environment before it gets pushed to production. The dashboard below demonstrates this capability - different releases are present in different environments, and it's easy to see where releases are in the lifecycle.

Jenkins is a Continuous Integration (CI) tool that can only do some parts of Continuous Delivery. Jenkins is commonly used to build and push images to a central repository. Octopus Deploy can interface with several different repositories and manage the deployment process. This separation of concerns allows Jenkins and Octopus Deploy to focus on what they're best at, enabling happier deployments.



Azure	Development	Test	Staging	Production
MySQL Helm Chart	✓ 2021.11.23.0 Nov 23, 2021 7:30 PM	✓ 2021.02.09.0 Feb 10, 2021 4:37 AM	✓ 2021.02.09.0 Feb 10, 2021 4:38 AM	✓ 2021.02.09.0 Feb 10, 2021 4:39 AM
Octo Pet Shop - Raw YAML	✓ 2021.11.23.0 Nov 23, 2021 7:43 PM	✓ 2021.06.28.0 Jul 7, 2021 5:51 AM	✓ 2021.06.28.0 Jul 7, 2021 6:00 AM	✓ 2021.02.11.0 Feb 11, 2021 11:56 AM

Conclusion

In this post, you built a web application using Jenkins, pushed it to the ECR, and used Octopus Deploy to manage the deployment to Kubernetes.

Octopus Deploy provides a dedicated dashboard to view deployments in their different stages. The dashboard highlights how Octopus Deploy supplements a CI tool like Jenkins.

Octopus Deploy supports all the major cloud providers, including Azure, Google, and Amazon. If you're not already using Octopus, you can start a free trial.

Check out our other posts about deploying with Jenkins, Kubernetes, and Octopus Deploy:

- Building a Docker image in Jenkinsfile and publishing to ECR
- Deploying to Amazon EKS with Docker and Jenkins

Try our free Jenkins Pipeline Generator tool to create a Pipeline file in Groovy syntax. It's everything you need to get your Pipeline project started.

Read the rest of our Continuous Integration series.

Happy deployments!

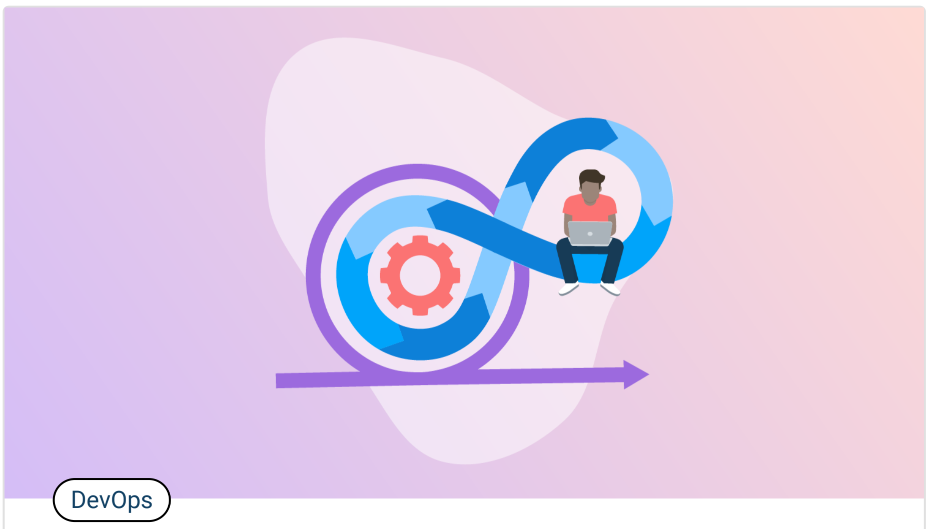
Tagged with: DevOps CI Series Continuous Integration Jenkins Kubernetes

Related posts

DevOps

Common mistakes in DevOps metrics

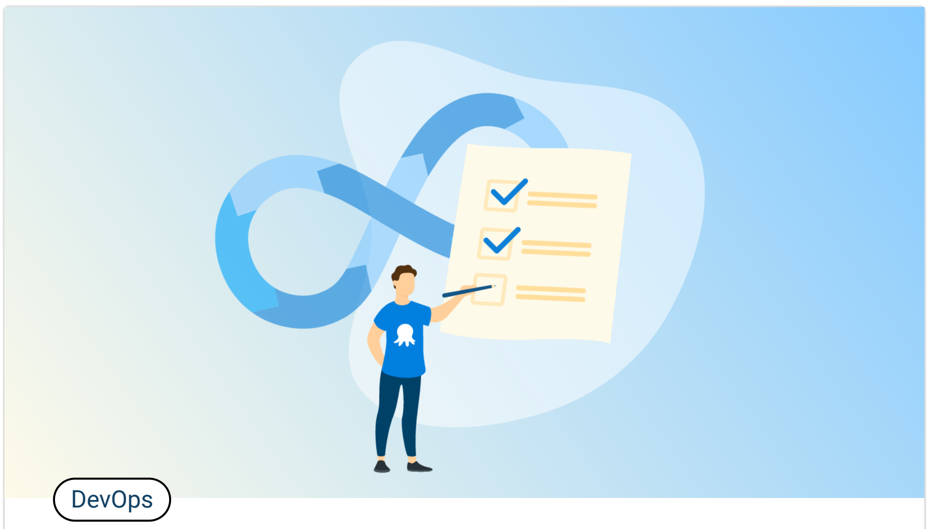
Steve Fenton
December 7, 2022 • 6 mins



DevOps

Comparing Lean, Agile, and Continuous Delivery

Steve Fenton
December 5, 2022 • 6 mins



DevOps

Best practices for CI/CD

Terence Wong
November 30, 2022 • 5 mins

Newsletter

Logged in as Terence Wong (terence.wong@octopus.com)

Join ~48,000 DevOps professionals and sign up for the latest Octopus news, events, and opinions. No spam. Unsubscribe at any time.

Subscribe

Your privacy is important to us. Read more in our **Privacy Policy**.