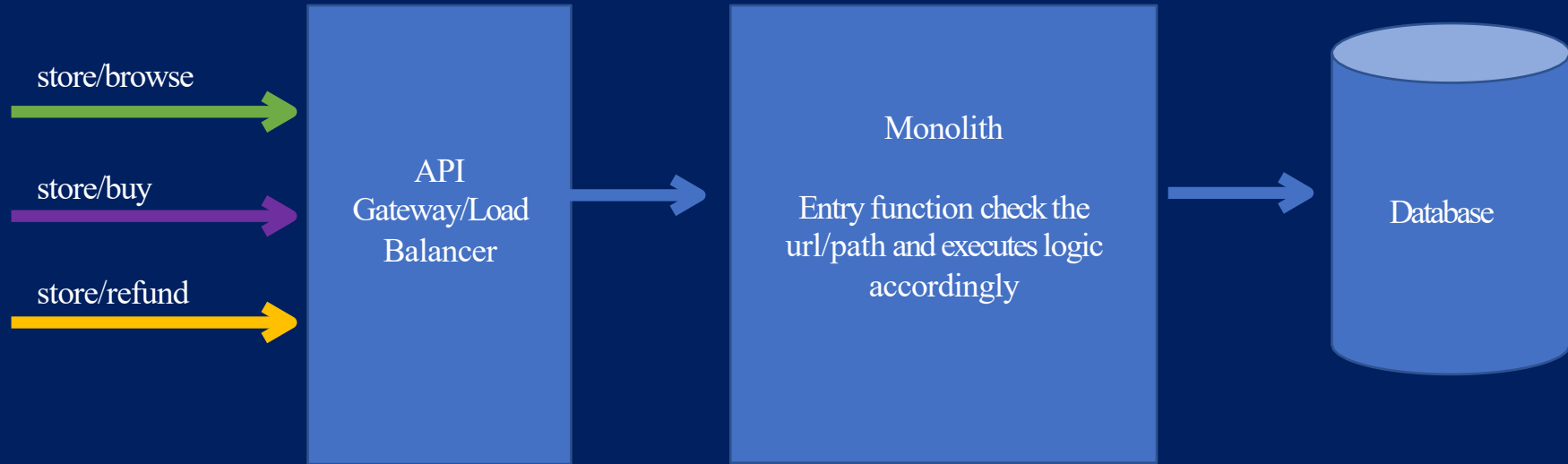


Week 5 : SOFTWARE DEVELOPMENT TOOLS AND ENVIRONMENTS

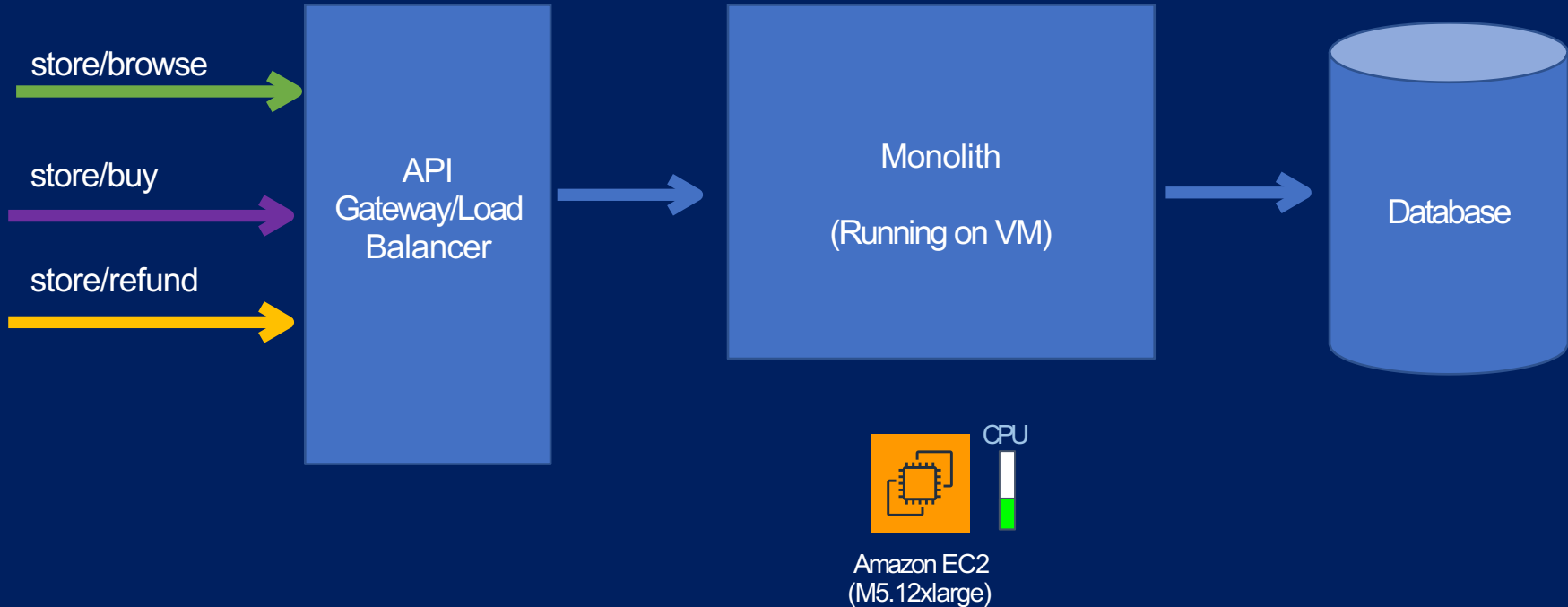
# DevOps – What and Why

---

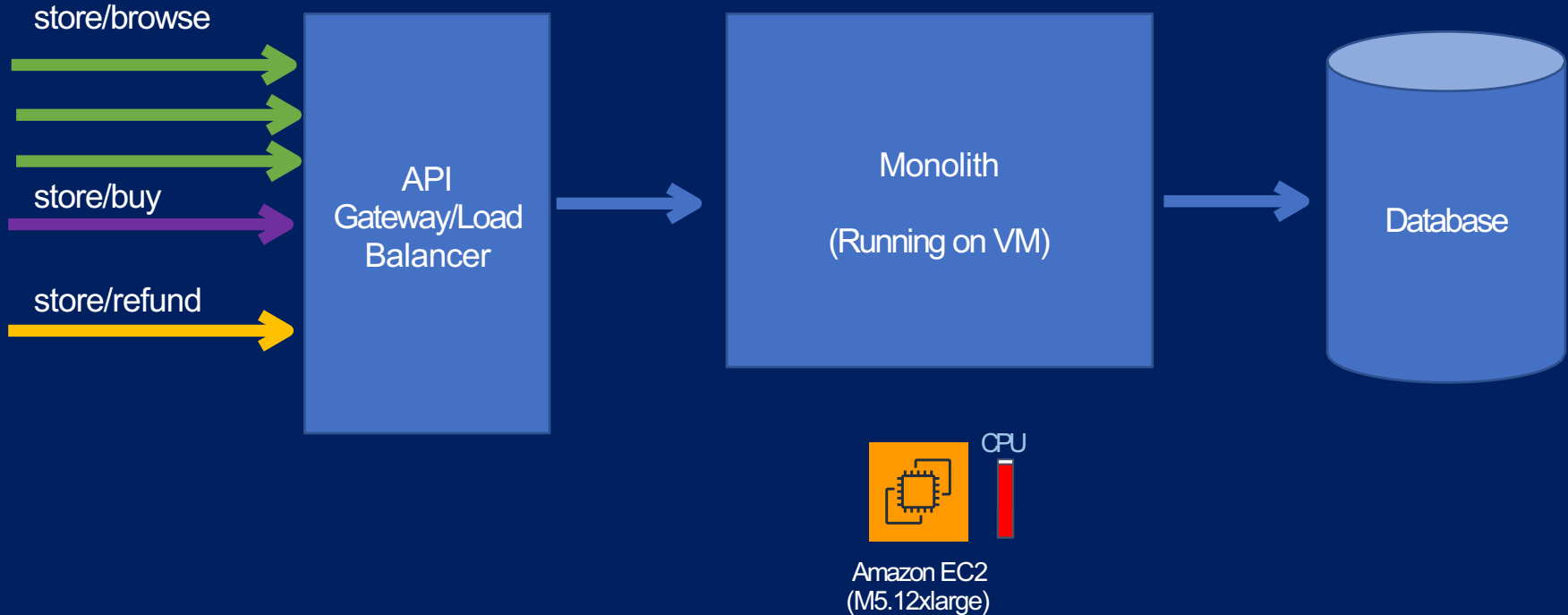
# Monolith



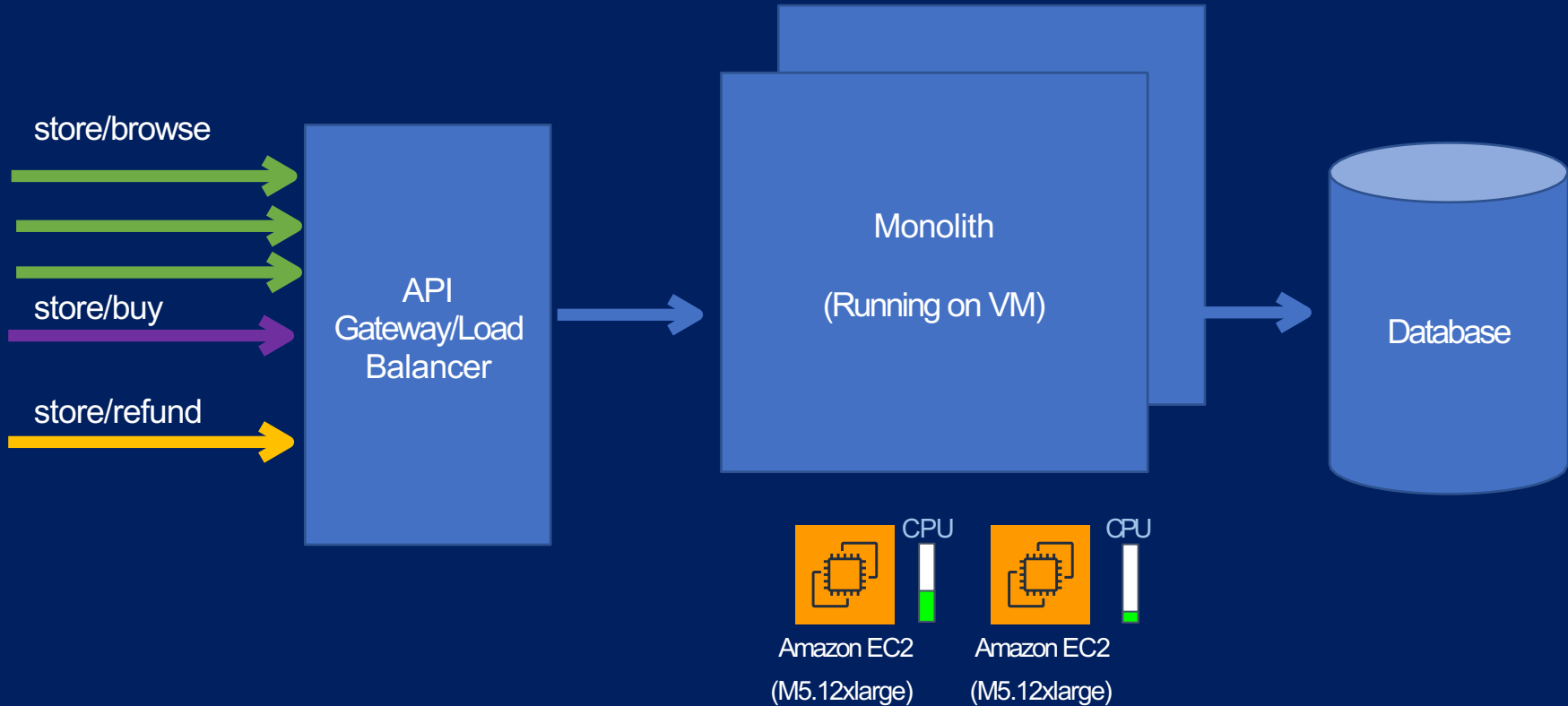
# Issue of Scaling



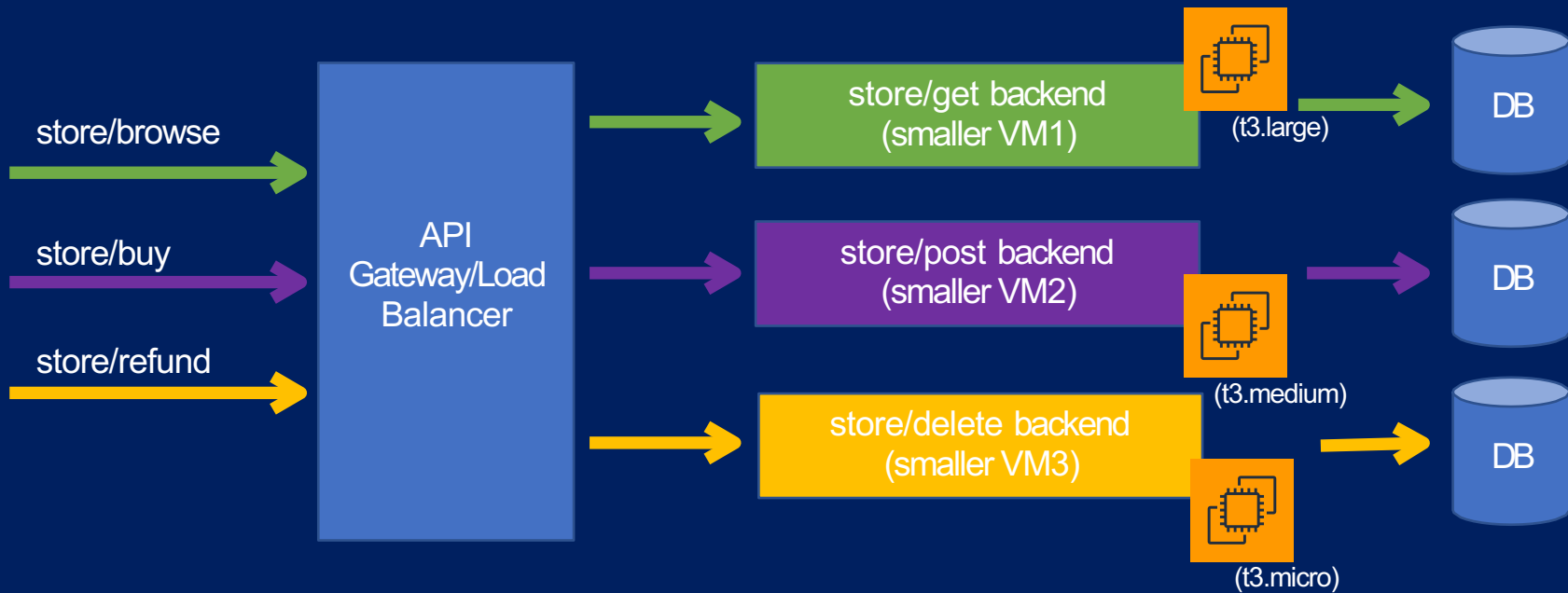
# Issue of Scaling



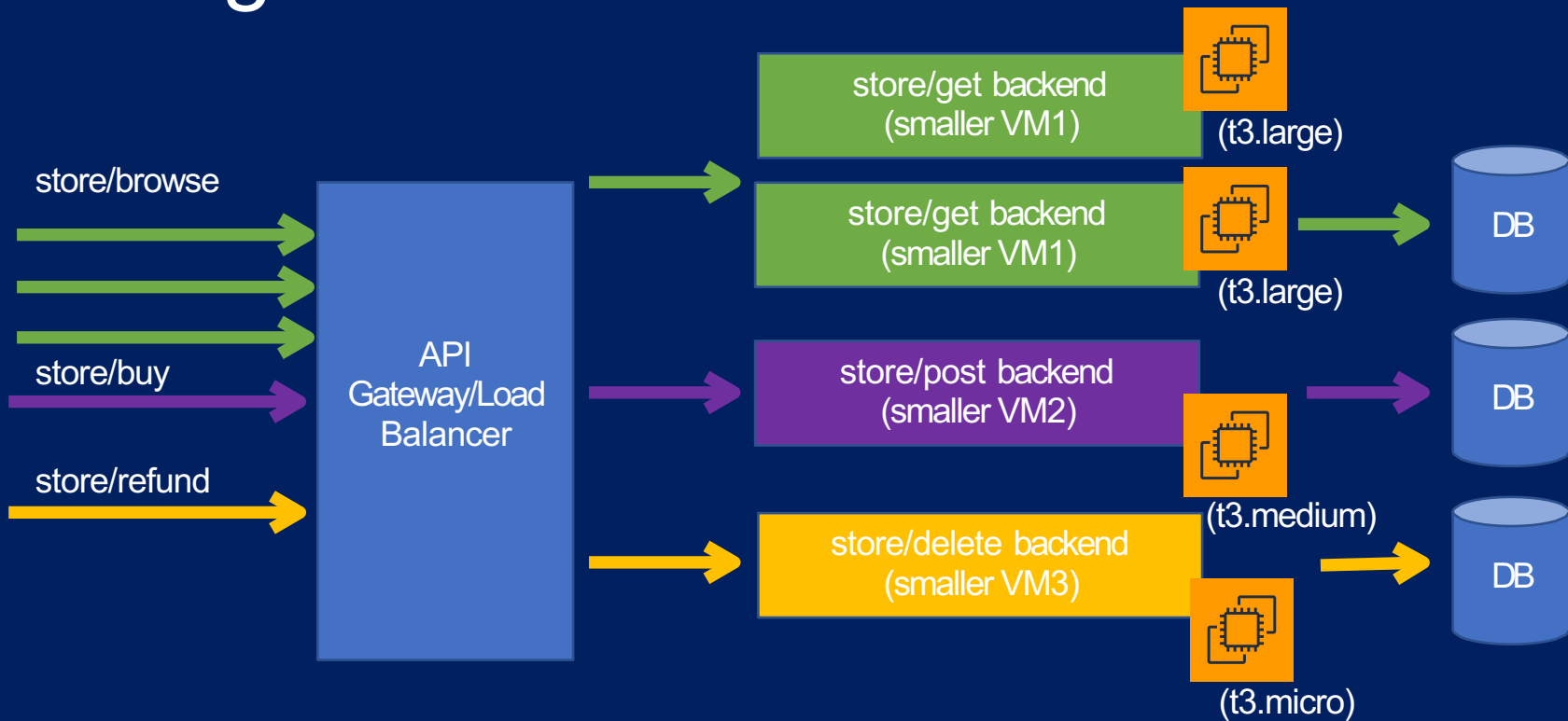
# Entire Monolith Need to Scale



# APIs in Microservice

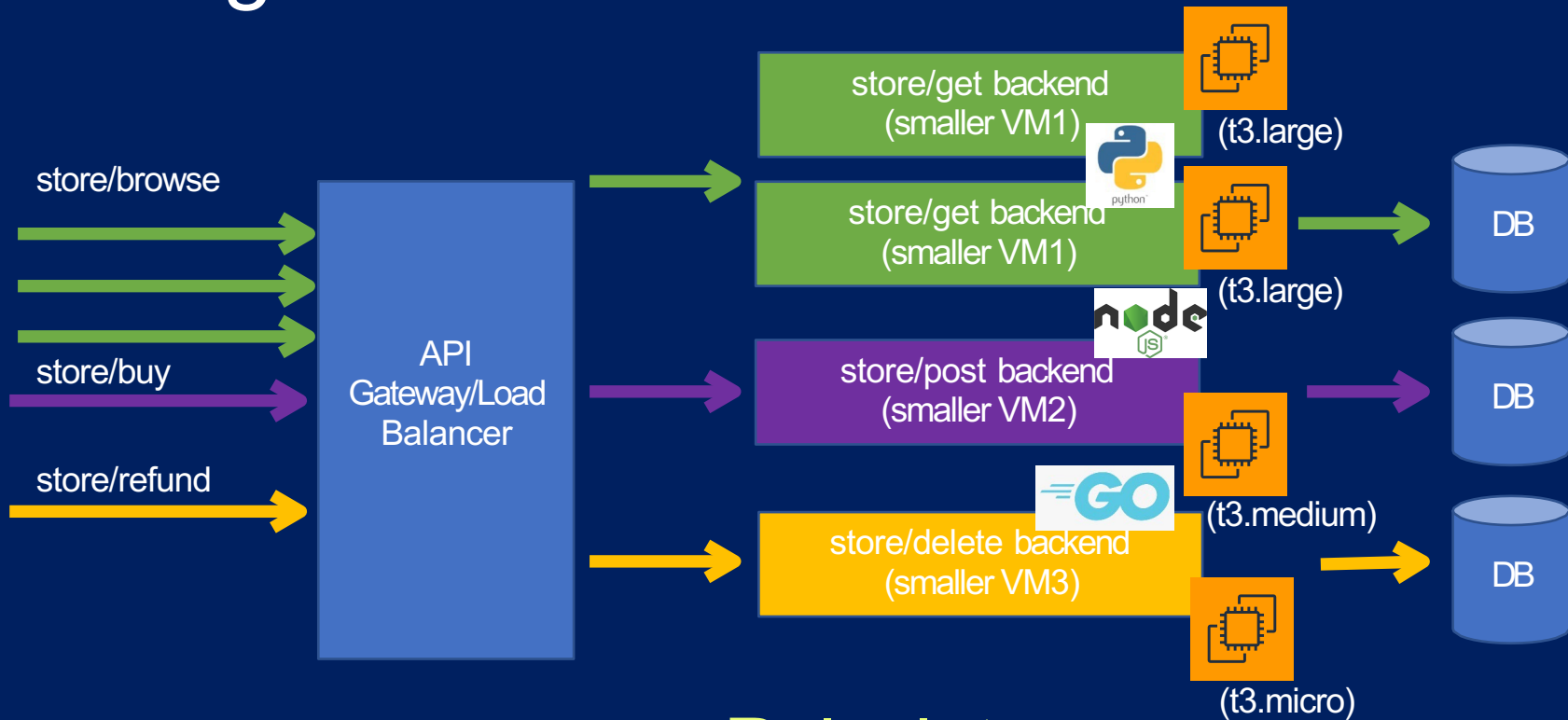


# Scaling APIs in Microservice





# Scaling APIs in Microservice



Polyglot

# World is Moving towards Microservice

- Microservices require frequent implementation



Code deployed every 11.7 seconds!



Delivery time reduced from  
hours to minutes

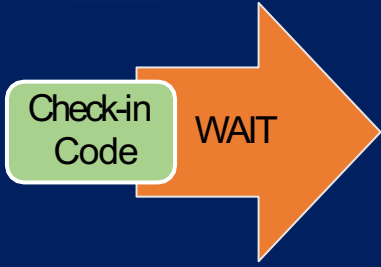


Code deployed thousand times per day

# Traditional Deployment

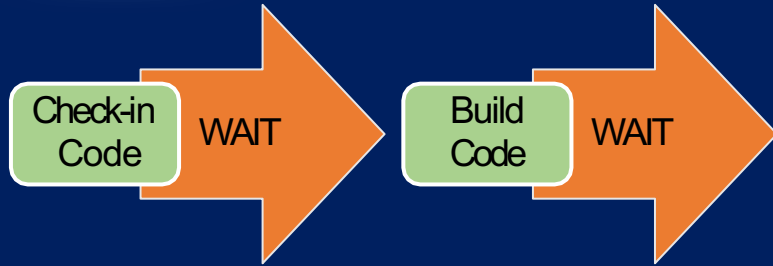


# Traditional Deployment



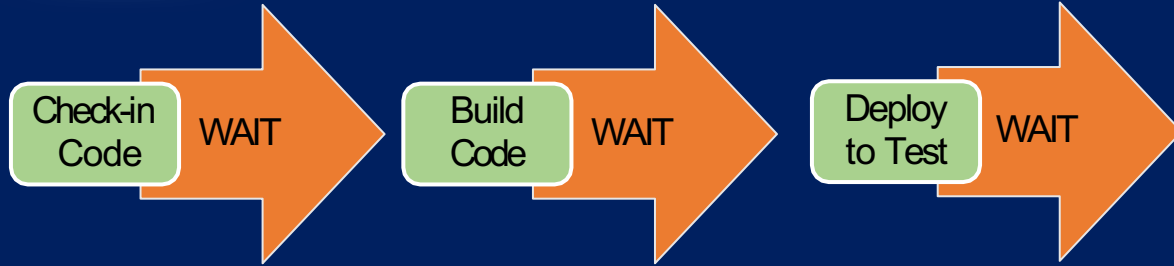
Hours/Days + Lot of Grief for Developer & Operations

# Traditional Deployment



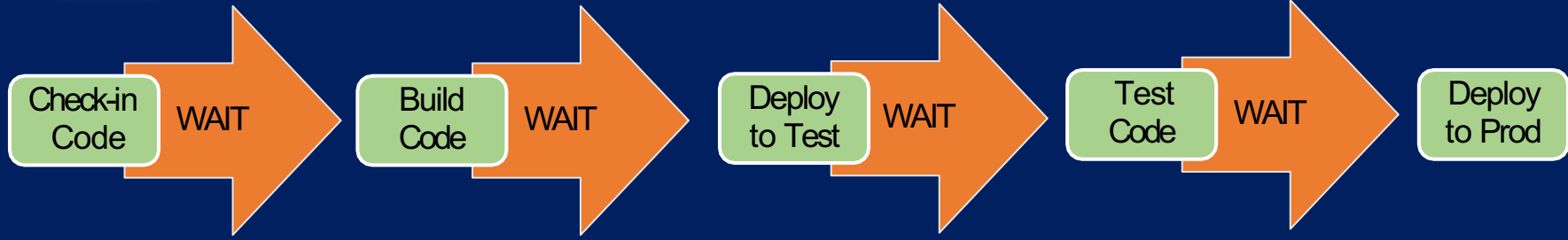
Hours/Days + Lot of Grief for Developer & Operations

# Traditional Deployment



Hours/Days + Lot of Grief for Developer & Operations

# Traditional Deployment



Hours/Days + Lot of Grief for Developer & Operations

# Traditional Deployment

When are you gonna deploy my code?

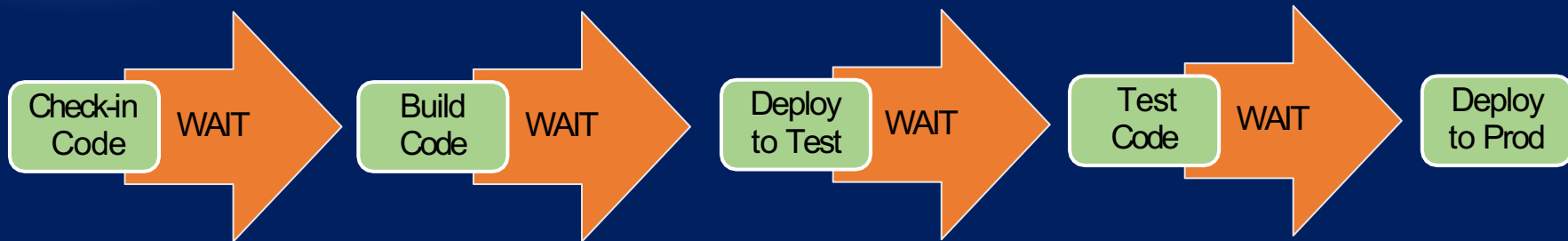


Developer

When you stop breaking my servers



Operations





SYSTEM →

I just  
wanna do  
cool stuff



# What is DevOps?

- Word “DevOps” coined in 2009 by Patrick Debois
- Combination of cultural philosophies, practices, and tools
  - Job market is based on tools!
- Development and Operations teams are no longer “siloesd”



# Traditional Deployment

When are you gonna deploy my code?



Developer

When you stop breaking my servers



Operations

Check-in  
Code

WAIT

Build  
Code

WAIT

Deploy  
to Test

WAIT

Test  
Code

WAIT

Deploy  
to Prod

# DevOps



Developer

Operations

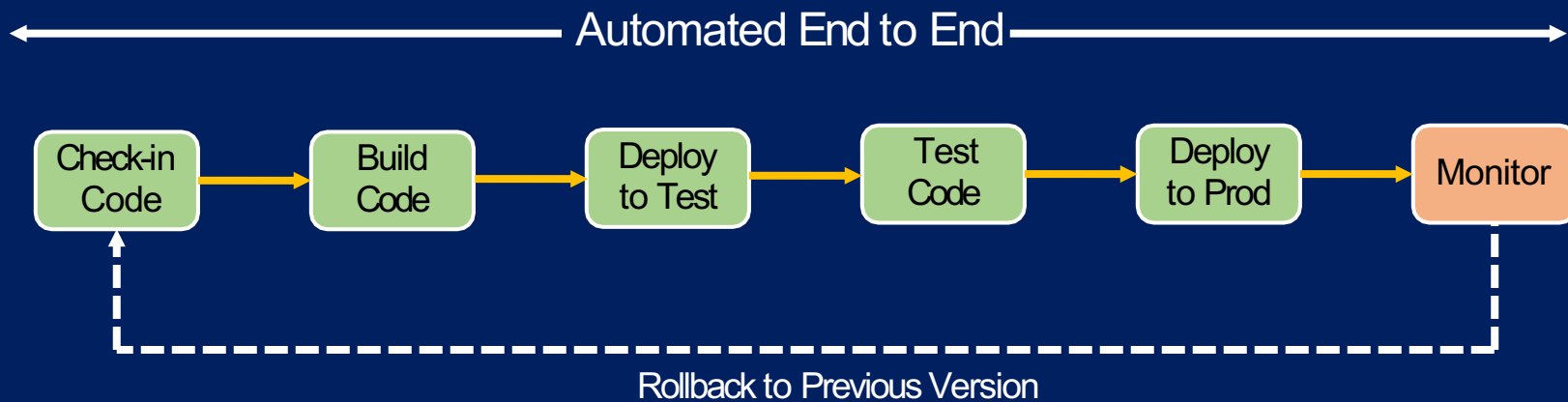
Same Team

← Automated End to End →



- Whole flow done in seconds!
- Easy to rollback in case of errors

# DevOps



# General DevOps Practices

- Automate everything!
- Deploy frequently rather than one mega deployment in months
- Codify every step - infrastructure, application and more
- Rome was not built in a day!

# DevOps Benefits

---

# DevOps Benefits



## Technical benefits

- Faster software delivery
- Faster problem remediation
- Easier to replicate best practices
- More time to innovate (rather than fix/maintain)



## Cultural benefits

- Improved communication and collaboration
- Greater professional opportunities
- Happier, more productive teams



# Why DevOps?

How long would it take your organization to deploy a change that involves a single line of code?

Can you do this on a repeatable reliable basis?



DevOps Vs Non-Devops organizations:

4x

Lower change  
failure rate

24x

Faster recovery times

200x

More frequent  
deployments

44%

More time spent  
on new features  
and code

Source: Puppet State of DevOps Report

# DevOps Challenges

---

# DevOps Challenges



## Challenges

- Continuously adapt to changing landscape
  - New tools
  - New processes and technologies
- Developers unwilling to provide support
- Takes months/years to ramp up
- Resistance to change

# DevOps Challenges



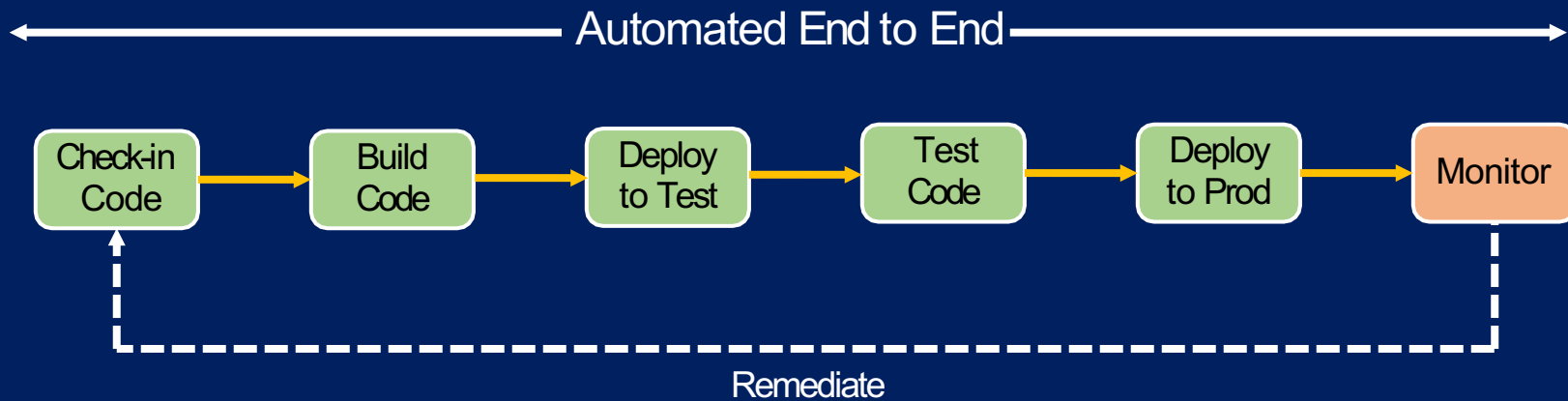
## Challenges

- Continuously adapt to changing landscape
  - Establish standard toolsets
  - CCoE provides templates with best practices
- Developers unwilling to provide support
  - Rotation, incentives
- Takes months/years to ramp up
  - Utilize vendor trainings, workshops
- Resistance to change
  - Cultural training

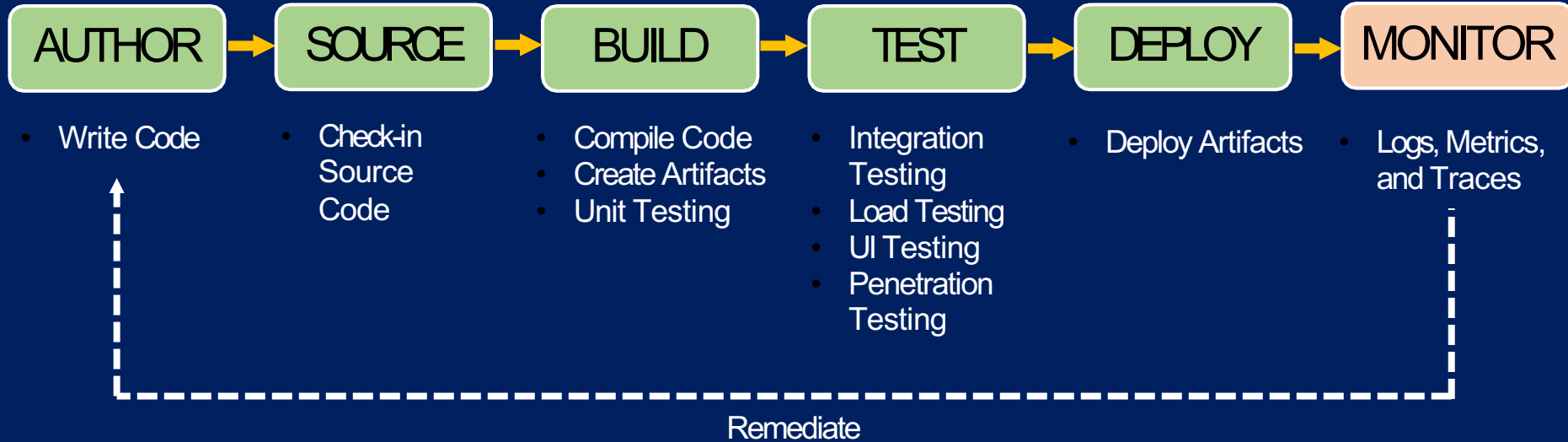
# CI vs CD vs CD

---

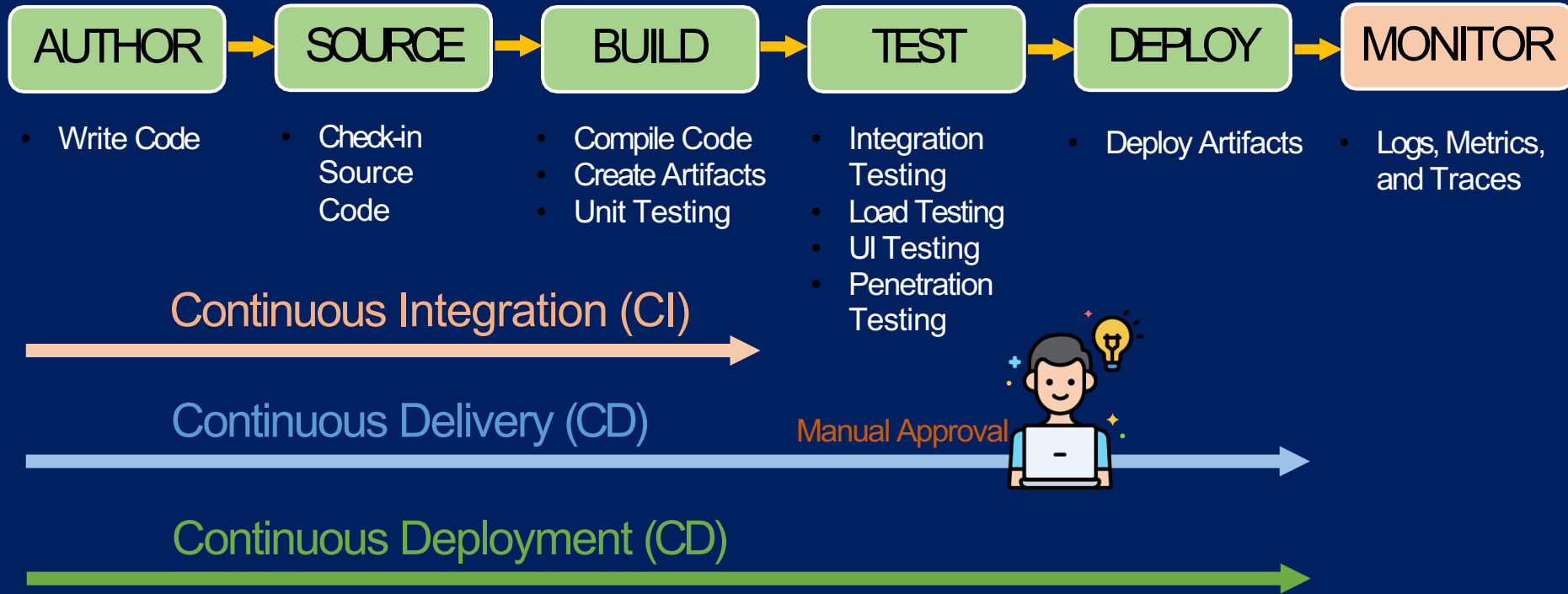
# DevOps



# DevOps Phases

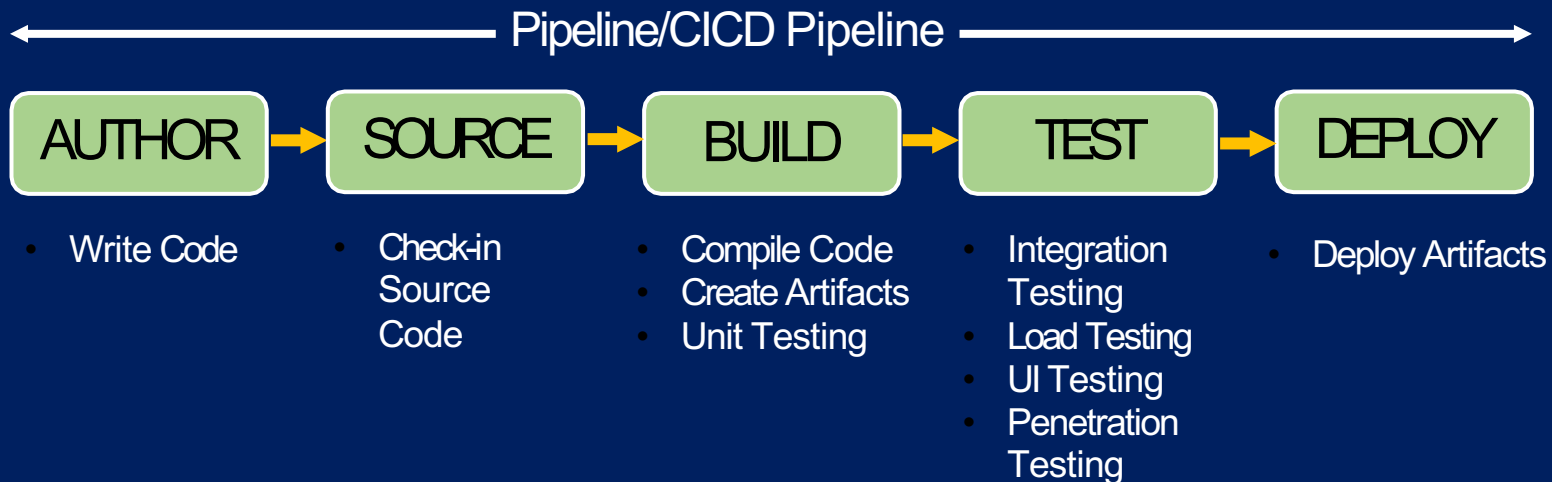


# CI vs CD vs CD

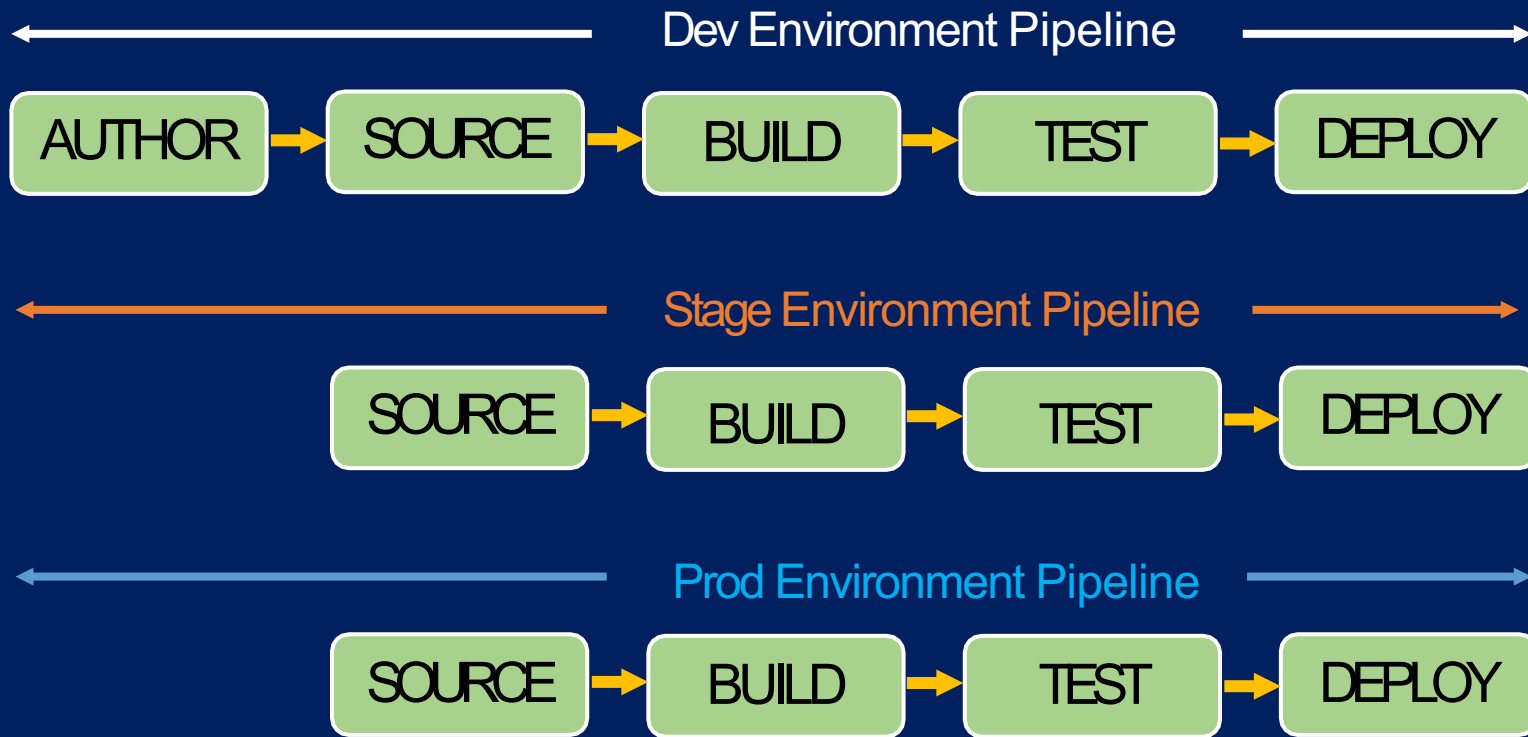




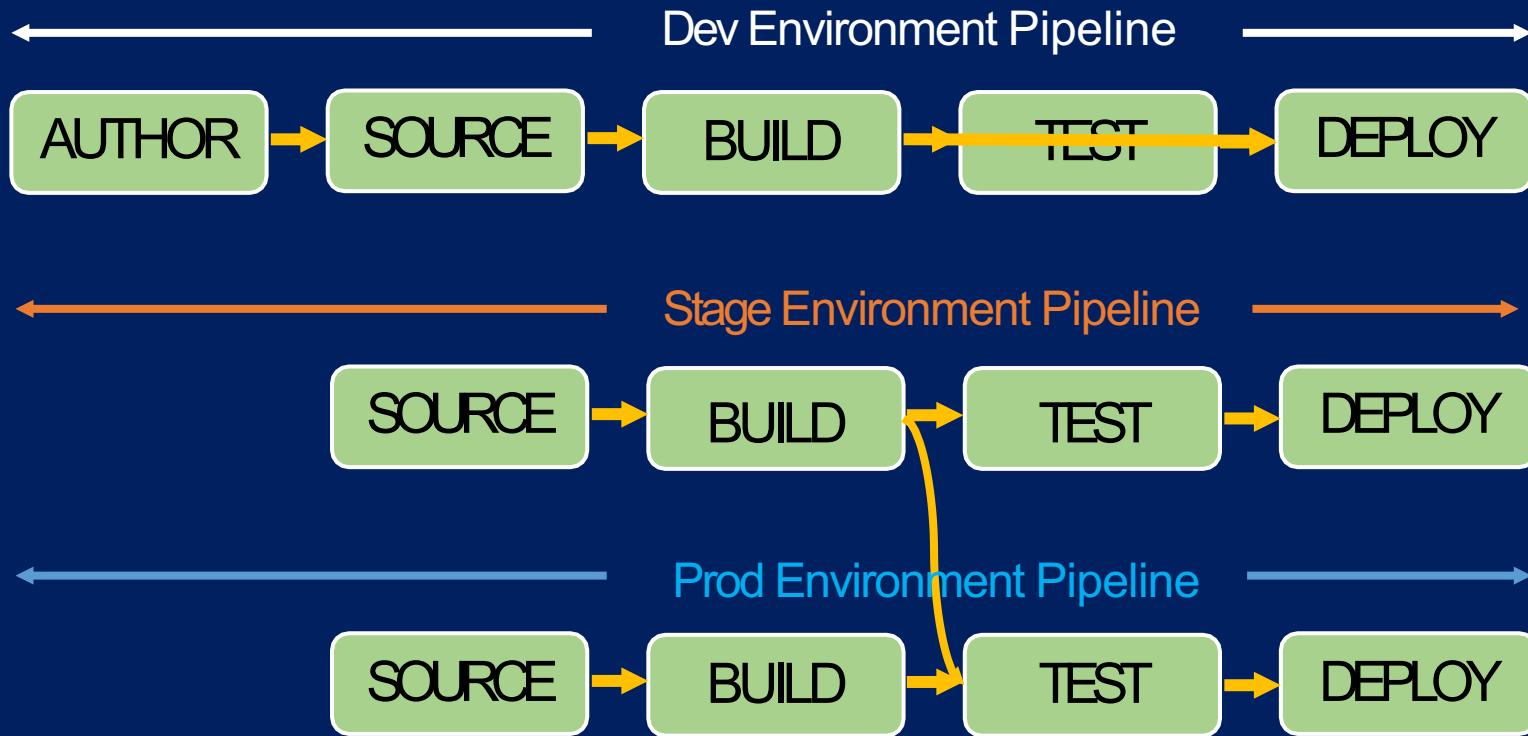
# DevOps Phases



# DevOps Pipelines



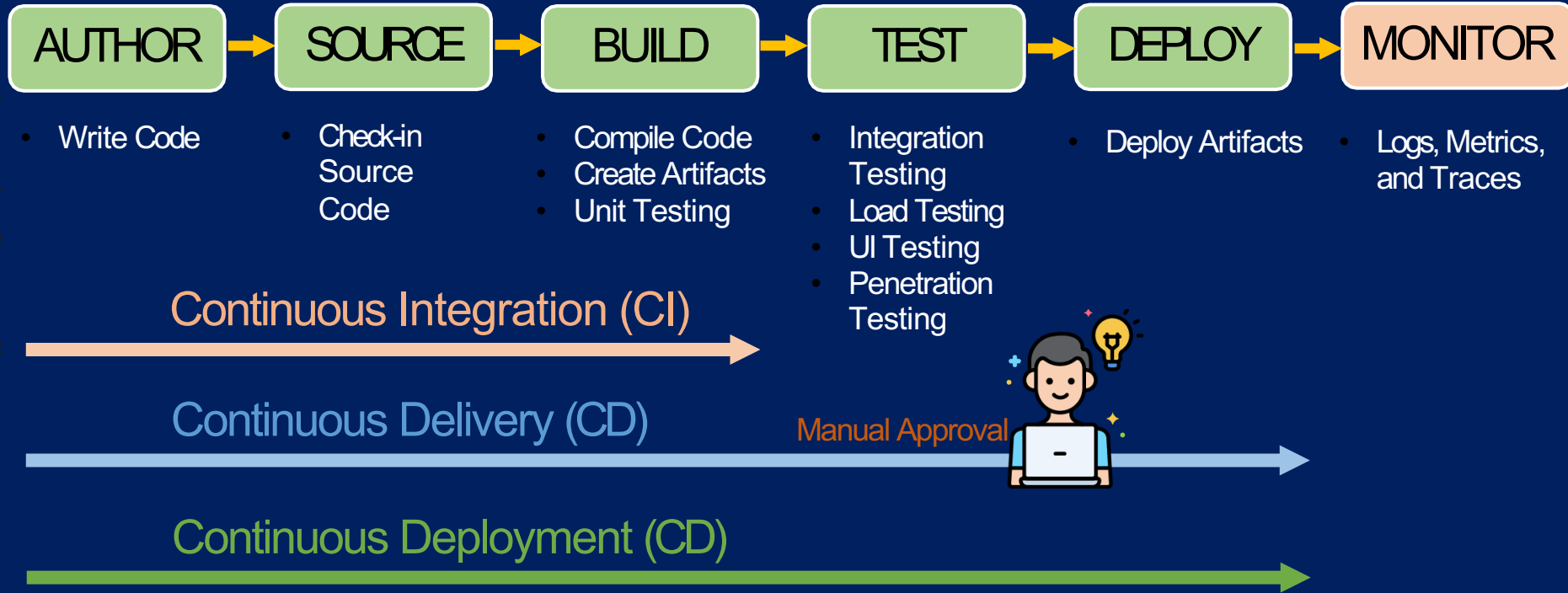
# DevOps Pipelines



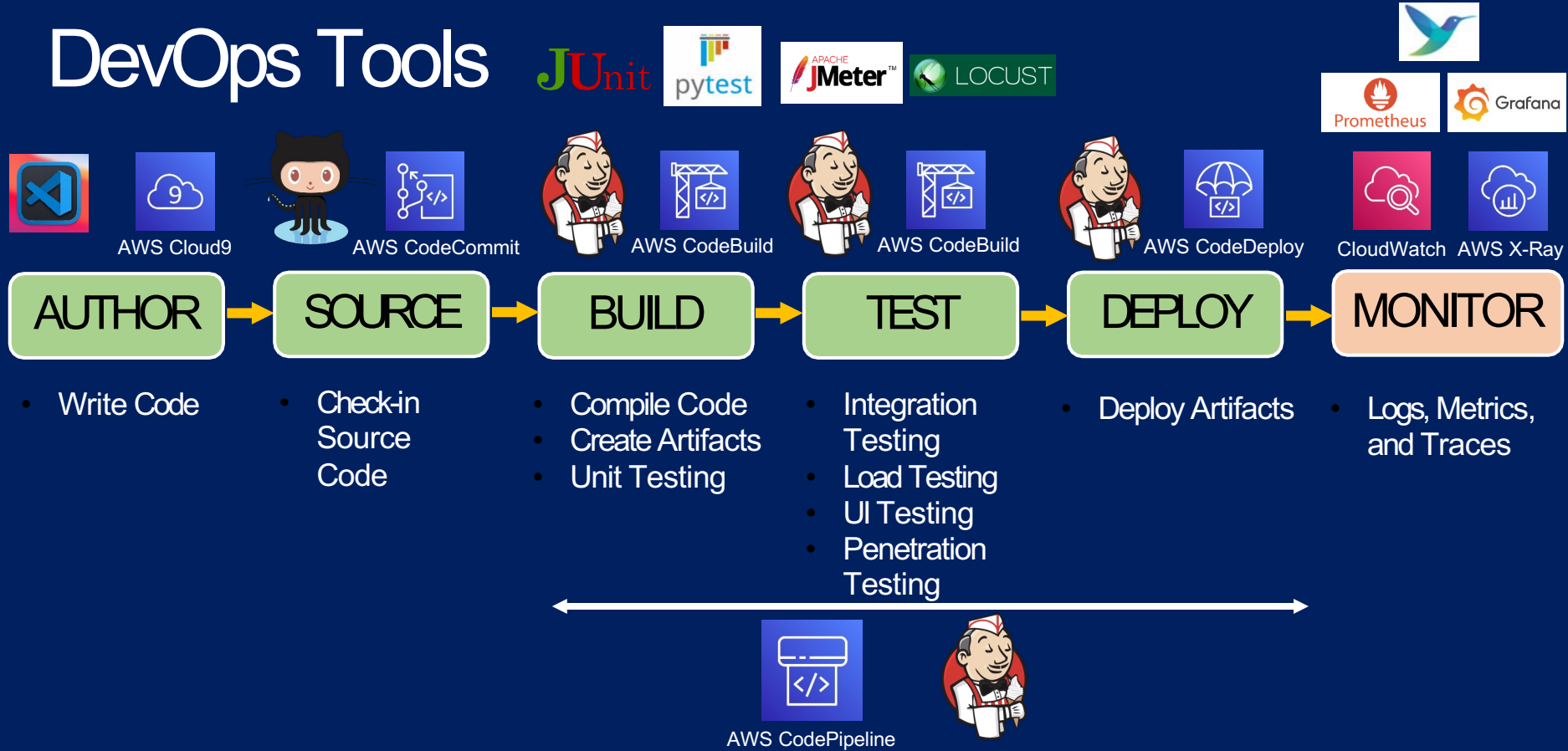
# DevOps Tools

---

# DevOps Phases



# DevOps Tools



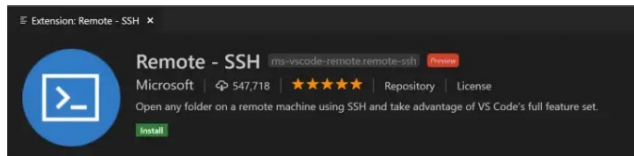
# Using VSCode remotely on an EC2 instance

# Using VSCode remotely on an EC2 instance

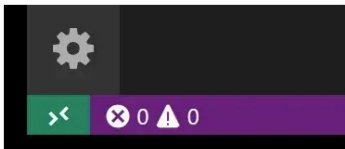
## Install the Remote SSH Extension

If you haven't already, download and install VSCode for your OS from [here](#).

You can then search for the extension "Remote-SSH" in the VSCode marketplace.

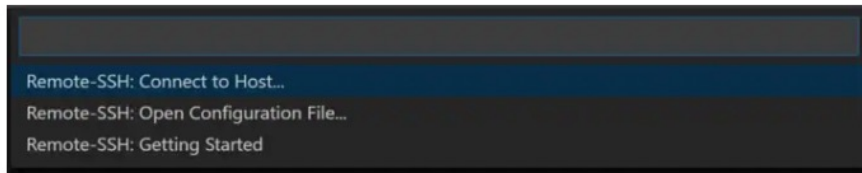


Once Installed, you should see a new Status bar item at the far left.

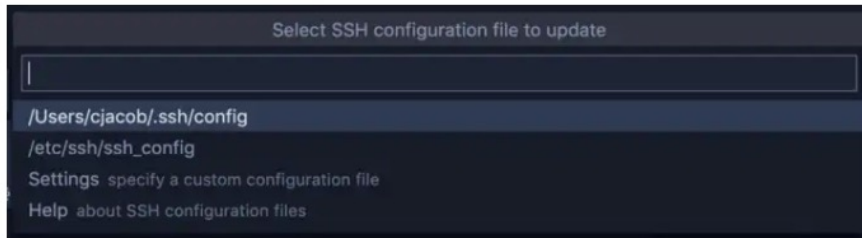


The status item can be used to quickly open the Remote SSH settings. Click on the status item.

The status item can be used to quickly open the Remote SSH settings. Click on the status item.



Open The Configuration file



<https://medium.com/@christyjacob4/using-vscode-remotely-on-an-ec2-instance-7822c4032cff>



# Using VSCode remotely on an EC2

```
1 Host aws-ec2
2   HostName ec2-44-229-243-8.us-west-2.compute.amazonaws.com
3   User ubuntu
4   IdentityFile ~/.aws-key/test-key-pair.pem
```

config hosted with ❤️ by GitHub

[view raw](#)

- **Host** (aws-ec2) is just a name that will appear in VS Code. It can be any name.
- **HostName** is the server's host or IP.
- **User** is the Ubuntu username.
- **IdentityFile** is the path to the private key.

To obtain the HostName and User for your instance, navigate to your EC2 console. **Right Click** on an instance > **Connect**. This will open up a dialog like

Example:

```
ssh -i "test-key-pair.pem" ubuntu@ec2-44-229-243-8.us-west-2.compute.amazonaws.com
```

# Jenkins on AWS

Jenkins is an open-source automation server that integrates with a number of AWS Services, including: AWS CodeCommit, AWS CodeDeploy, Amazon EC2 Spot, and Amazon EC2 Fleet. You can use Amazon Elastic Compute Cloud (Amazon EC2) to deploy a Jenkins application on AWS.

This tutorial walks you through the process of deploying a Jenkins application. You will launch an EC2 instance, install Jenkins on that instance, and configure Jenkins to automatically spin up Jenkins agents if build abilities need to be augmented on the instance.

In this tutorial, you will perform the following steps:

1. [Prerequisites](#).
2. [Create a key pair](#) using Amazon EC2. If you already have one, you can skip to step 3.
3. [Create a security group](#) for your Amazon EC2 instance. If you already have one, you can skip to step 4.
4. [Launch an Amazon EC2 instance](#).
5. [Install and configure Jenkins](#).
6. [Clean up tutorial resources](#).

<https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkins-on-AWS/>

## Jenkins on Docker



<https://www.youtube.com/watch?v=QNZNfvrFBMo&t=664s>

<https://www.jenkins.io/doc/book/installing/docker/>

```
docker run -p 8080:8080 -p 50000:50000 --restart=on-failure jenkins/jenkins:its-jdk11
```

## **Jenkins & GitHub Integration with Selenium Python**

<https://www.youtube.com/watch?v=h2Abne2jljM&t=248s>



## Declarative Pipeline fundamentals

In Declarative Pipeline syntax, the `pipeline` block defines all the work done throughout your entire Pipeline.

*Jenkinsfile (Declarative Pipeline)*

```
pipeline {  
  agent any ❶  
  stages {  
    stage('Build') { ❷  
      steps {  
        // ❸  
      }  
    }  
    stage('Test') { ❹  
      steps {  
        // ❺  
      }  
    }  
    stage('Deploy') { ❻  
      steps {  
        // ❼  
      }  
    }  
  }  
}
```

- ❶ Execute this Pipeline or any of its stages, on any available agent.
- ❷ Defines the "Build" stage.
- ❸ Perform some steps related to the "Build" stage.
- ❹ Defines the "Test" stage.
- ❺ Perform some steps related to the "Test" stage.
- ❻ Defines the "Deploy" stage.
- ❼ Perform some steps related to the "Deploy" stage.

<https://www.jenkins.io/doc/book/pipeline/>

# Python

*Jenkinsfile (Declarative Pipeline)*

```
/* Requires the Docker Pipeline plugin */
pipeline {
  agent { docker { image 'python:3.10.7-alpine' } }
  stages {
    stage('build') {
      steps {
        sh 'python --version'
      }
    }
  }
}
```

# Node.js / JavaScript

*Jenkinsfile (Declarative Pipeline)*

```
/* Requires the Docker Pipeline plugin */
pipeline {
  agent { docker { image 'node:16.17.1-alpine' } }
  stages {
    stage('build') {
      steps {
        sh 'node --version'
      }
    }
  }
}
```

# Go

*Jenkinsfile (Declarative Pipeline)*

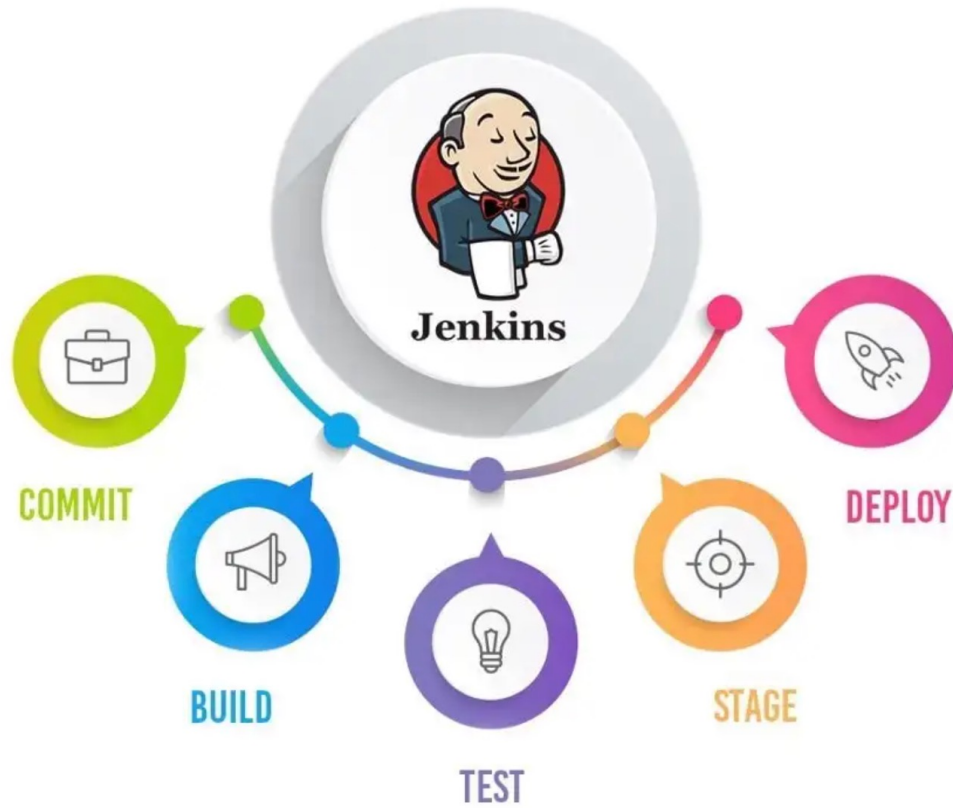
```
/* Requires the Docker Pipeline plugin */
pipeline {
  agent { docker { image 'golang:1.19.1-alpine' } }
  stages {
    stage('build') {
      steps {
        sh 'go version'
      }
    }
  }
}
```

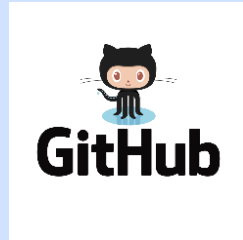
<https://www.jenkins.io/doc/pipeline/tour/hello-world/>







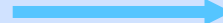
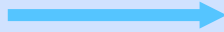




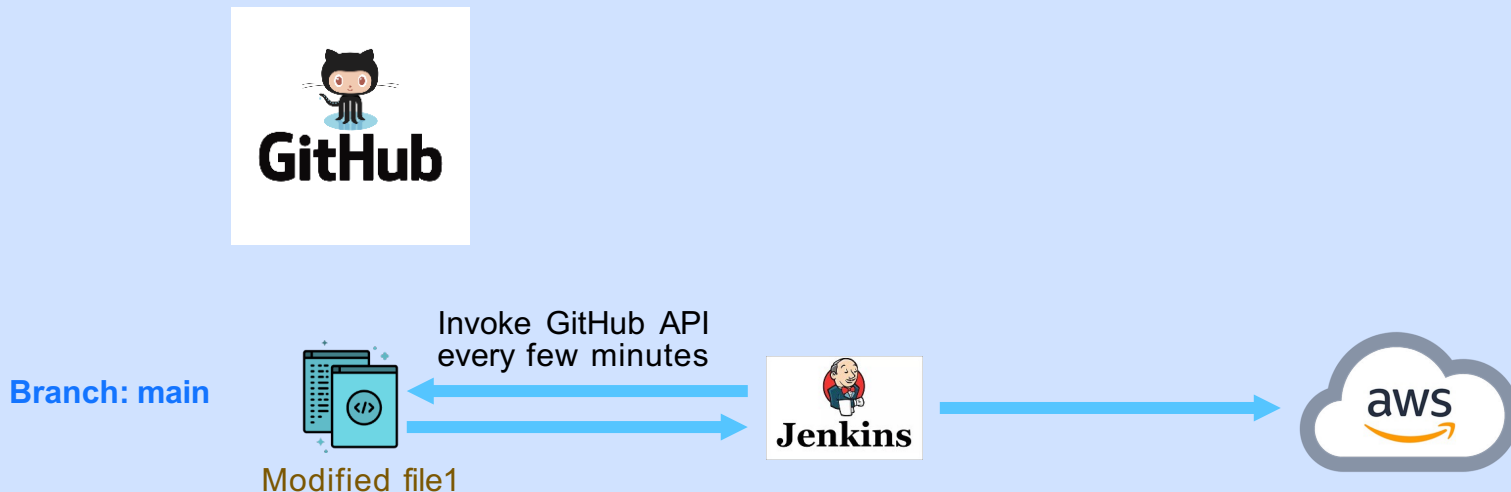
Branch: main



Modified file1

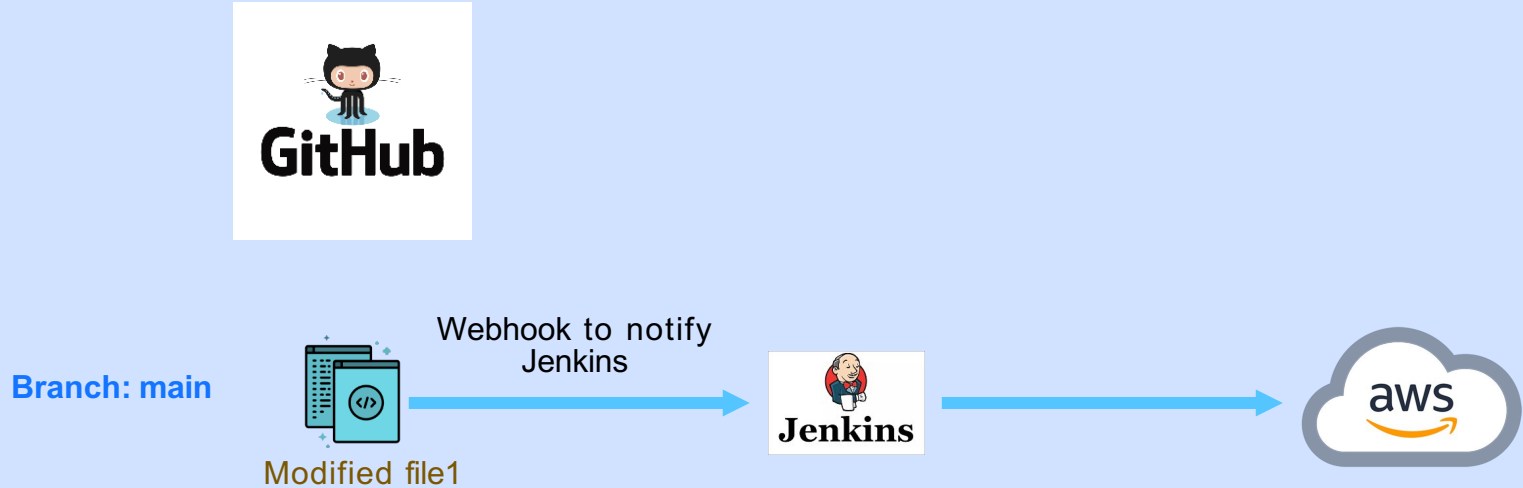


# Calling API



- Most of the times API will return stagnant data
- GitHub server will be bombarded
- Apps will exceed API limits

# Webhook



- GitHub will do a POST call to your app if repo changes
- Lightweight
- Realtime

# Implementation

