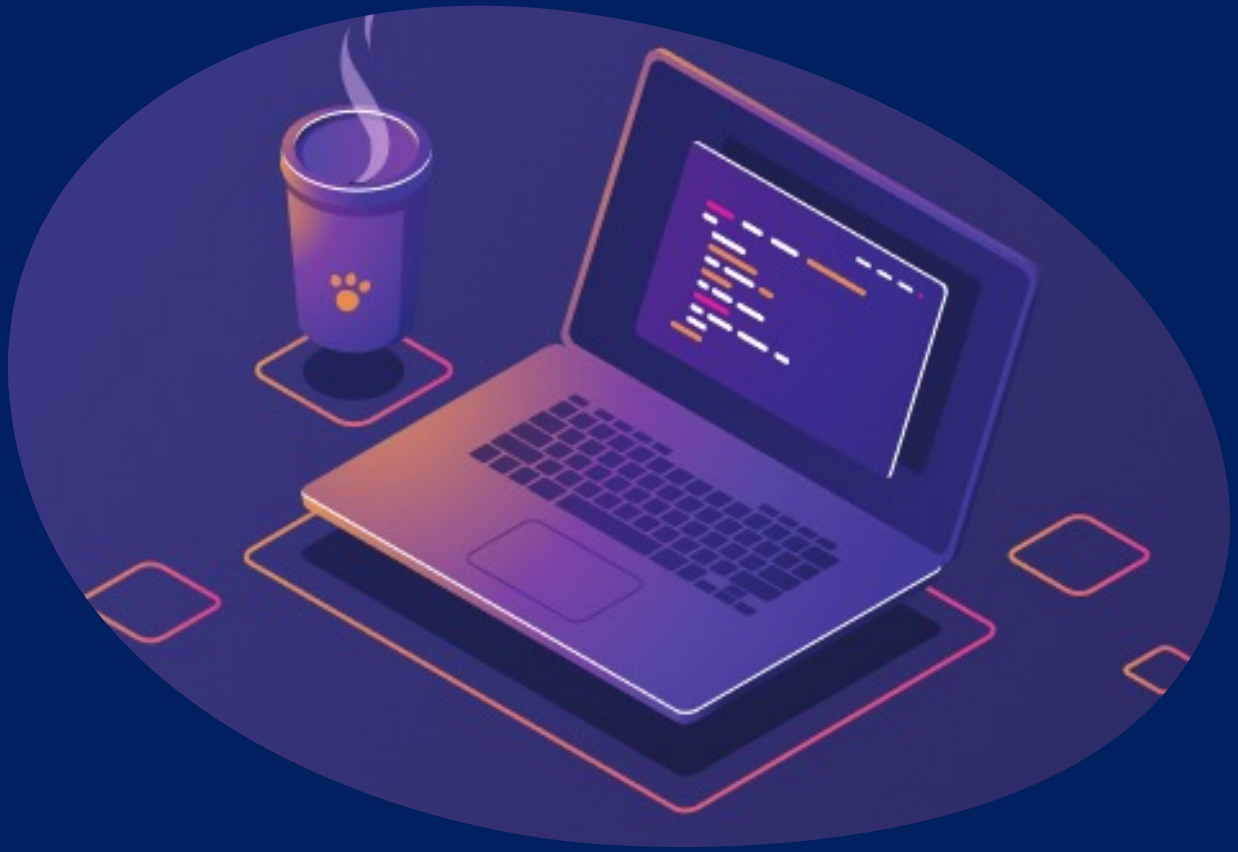
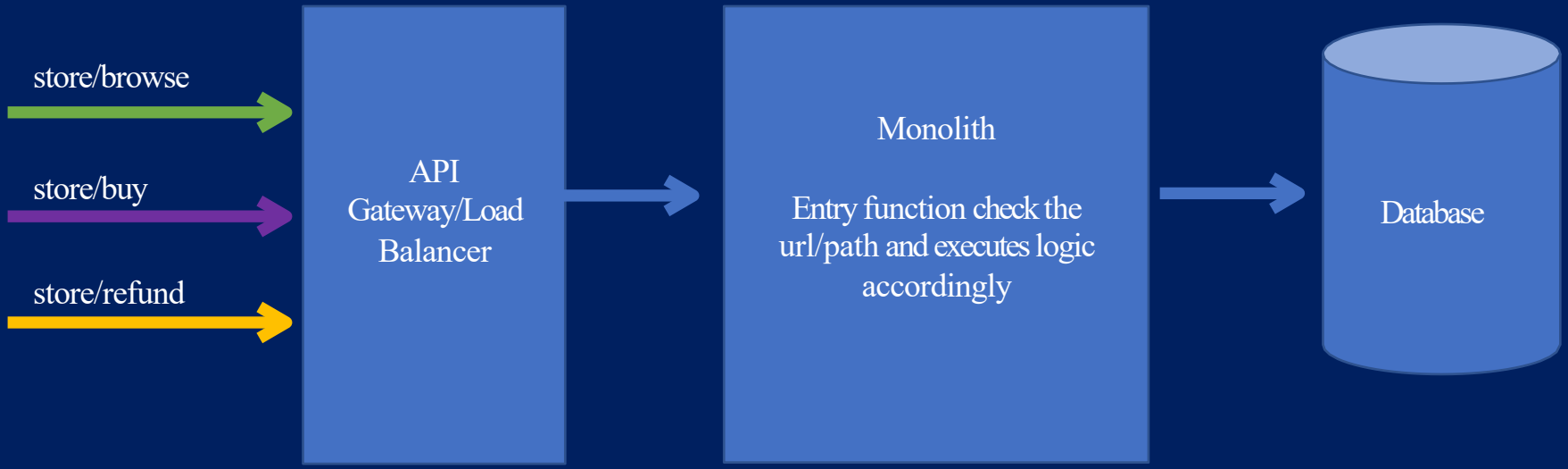


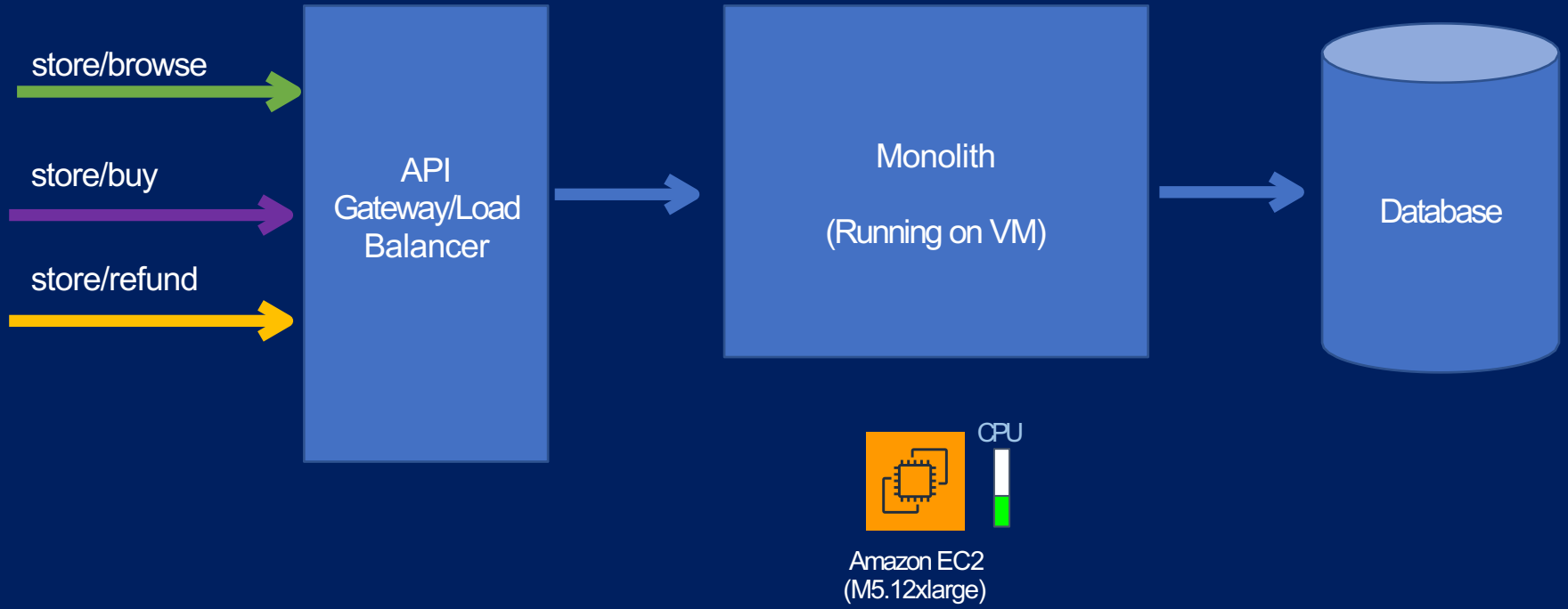
Week 5 : SOFTWARE DEVELOPMENT TOOLS AND ENVIRONMENTS

DevOps – What and Why

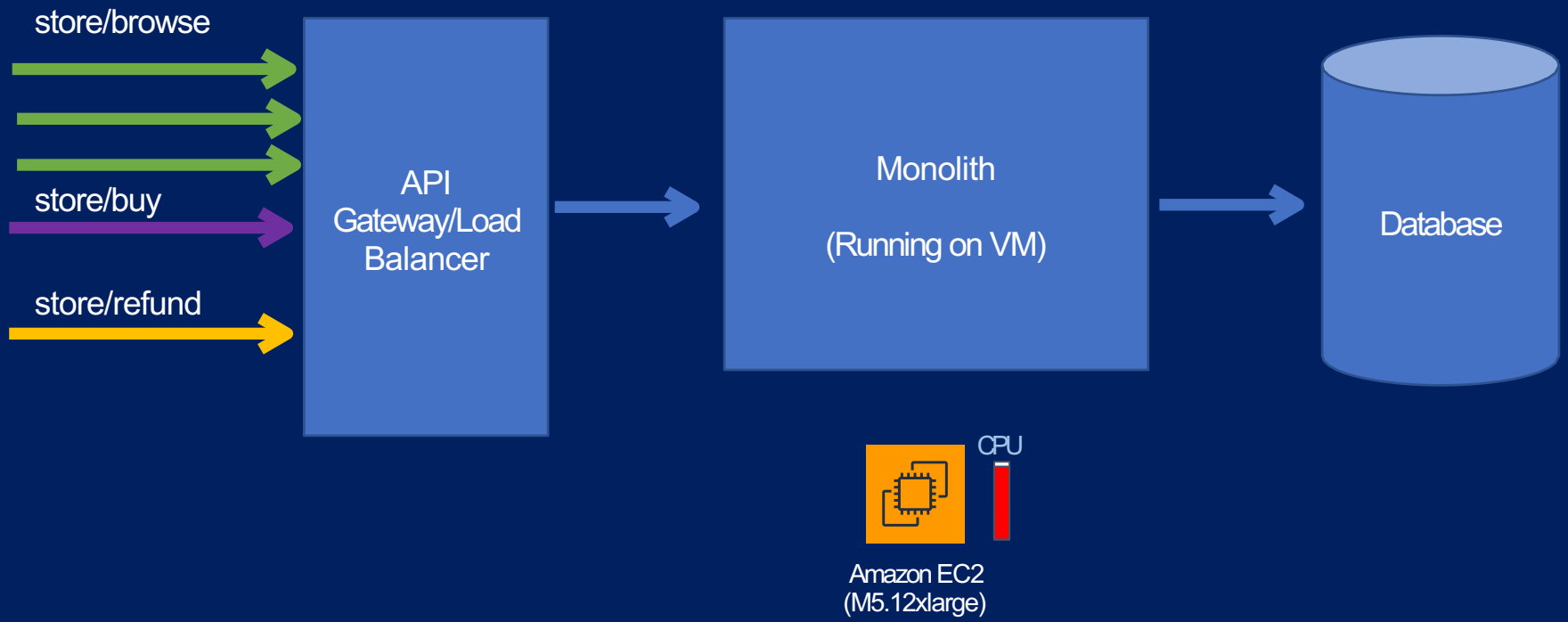
Monolith


store/browse
store/buy
store/refund
API Gateway/Load Balancer
Monolith
Entry function check the url/path and executes logic accordingly
Database

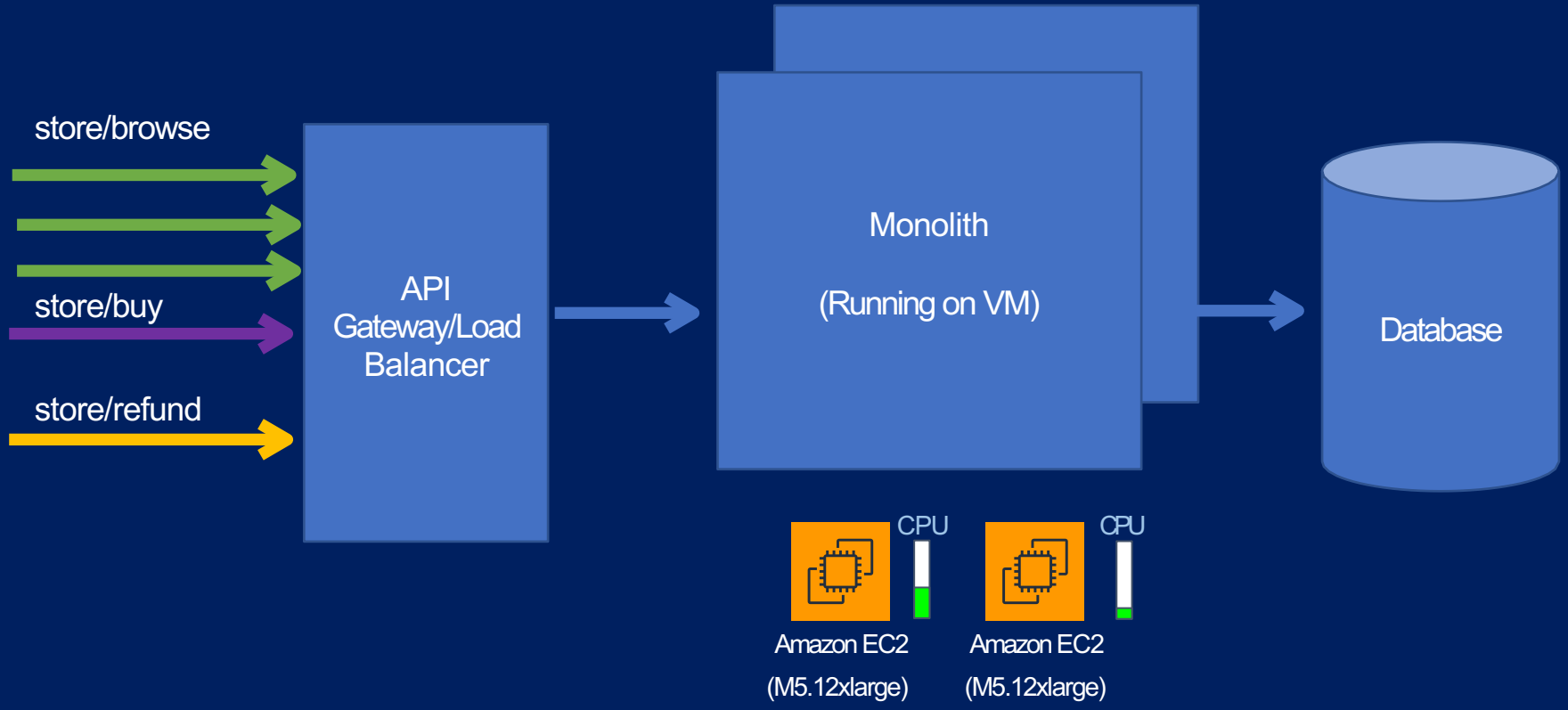
Issue of Scaling



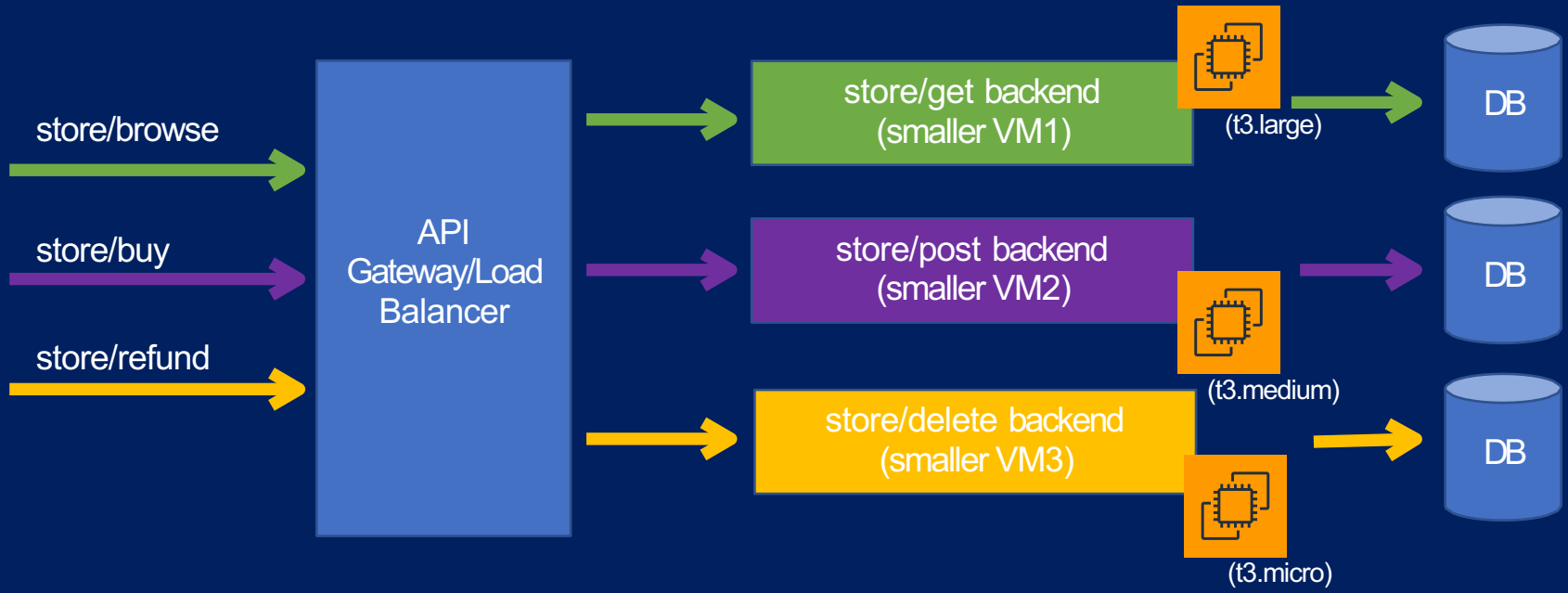
Issue of Scaling



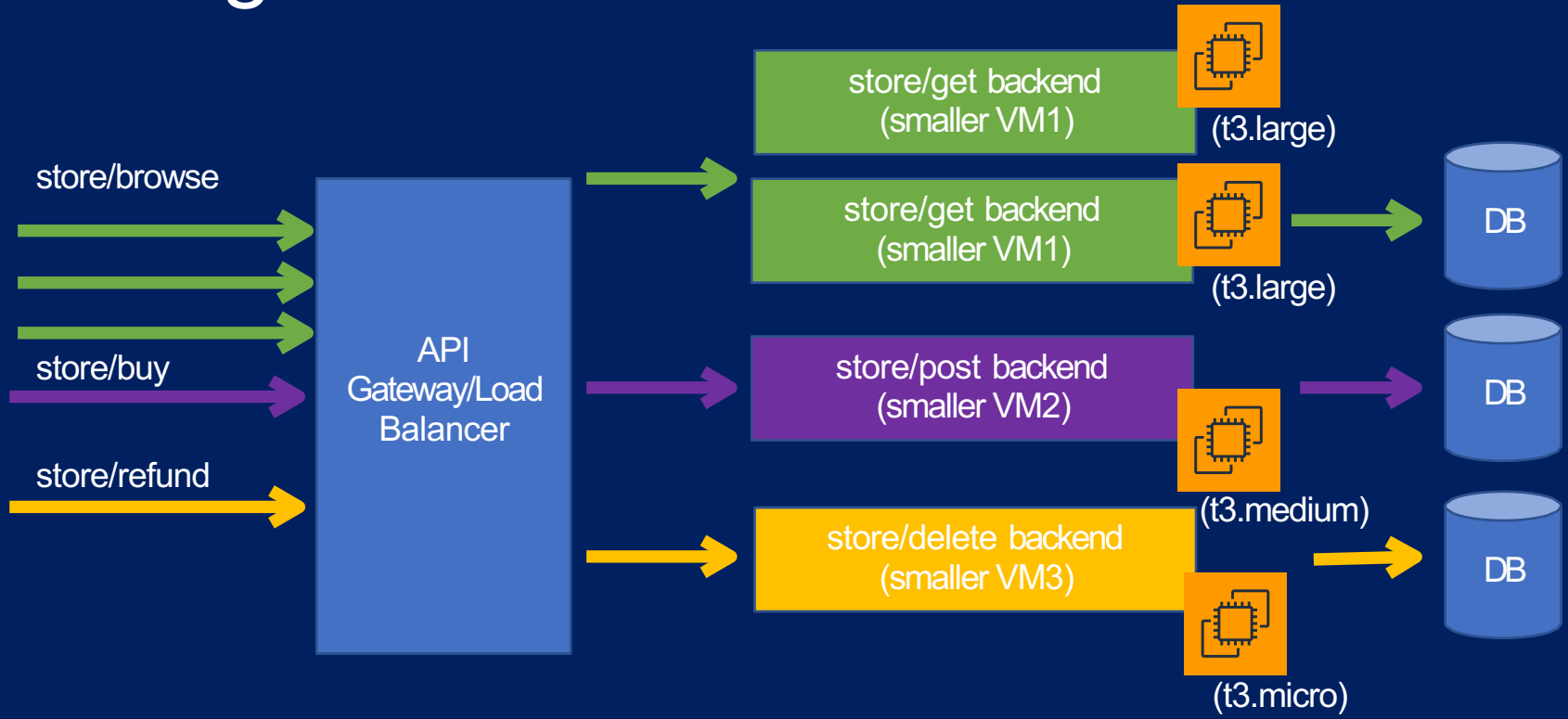
Entire Monolith Need to Scale



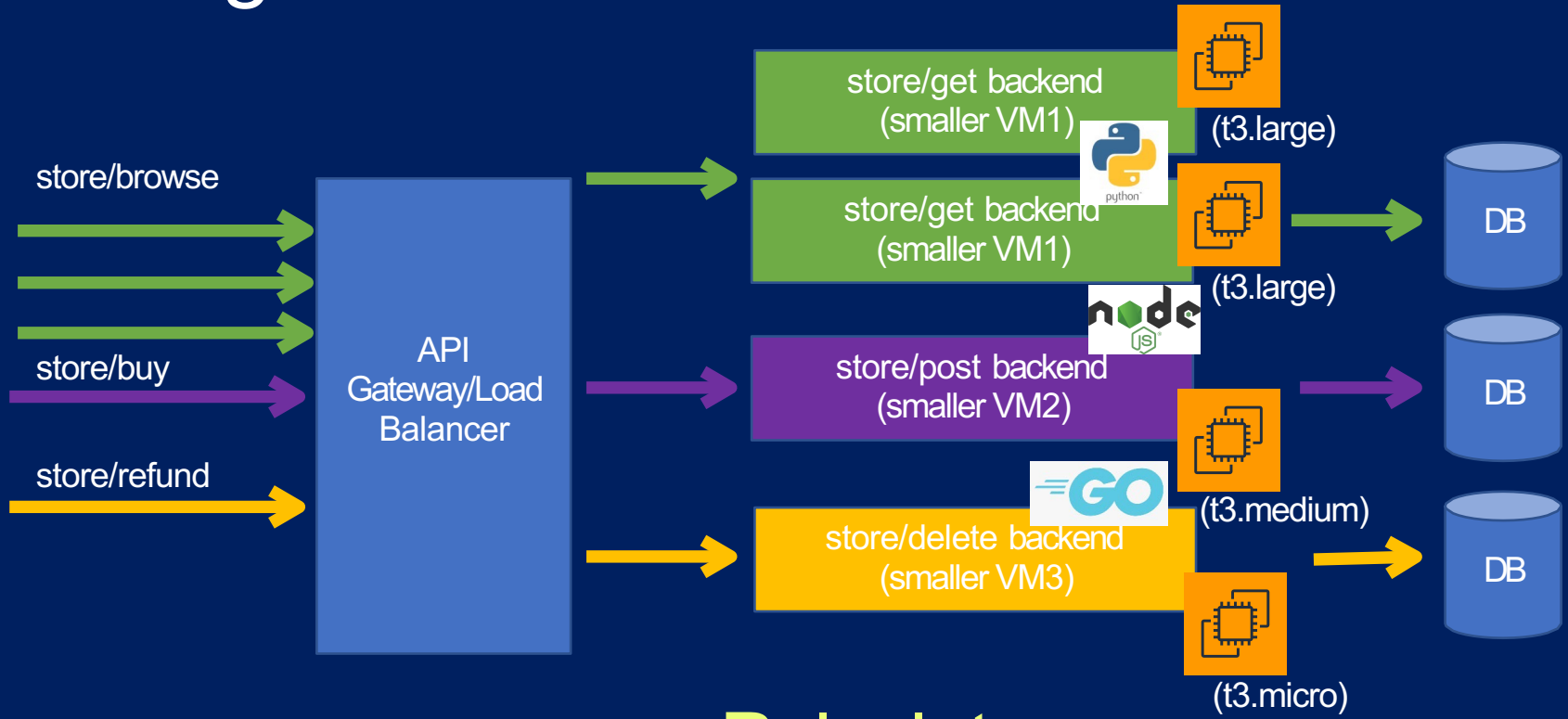
APIs in Microservice



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

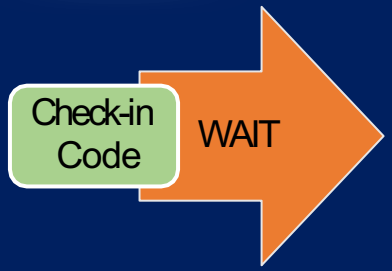
Developer

Operations

Traditional Deployment

Developer

Operations

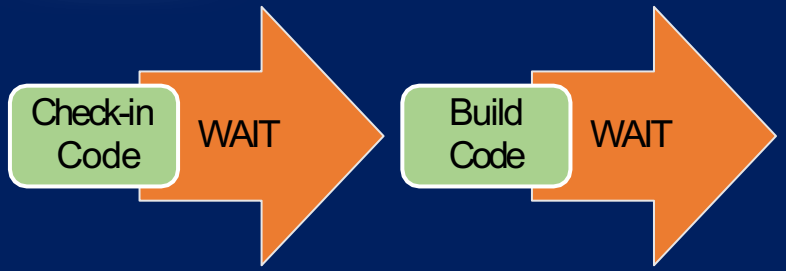


Hours/Days + Lot of Grief for Developer & Operations

Traditional Deployment

Developer

Operations

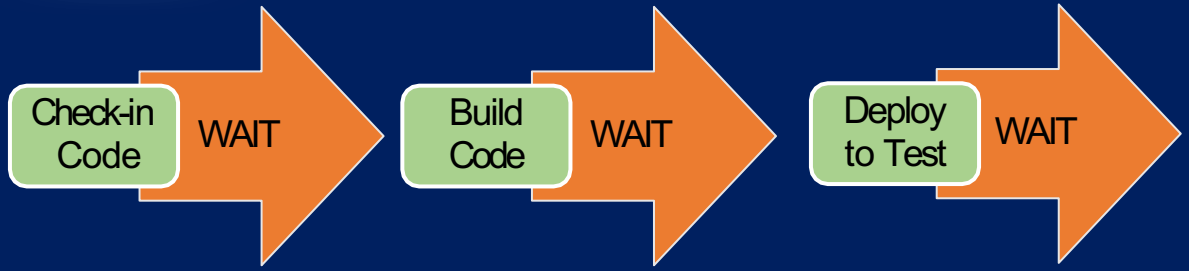


Hours/Days + Lot of Grief for Developer & Operations

Traditional Deployment

Developer

Operations

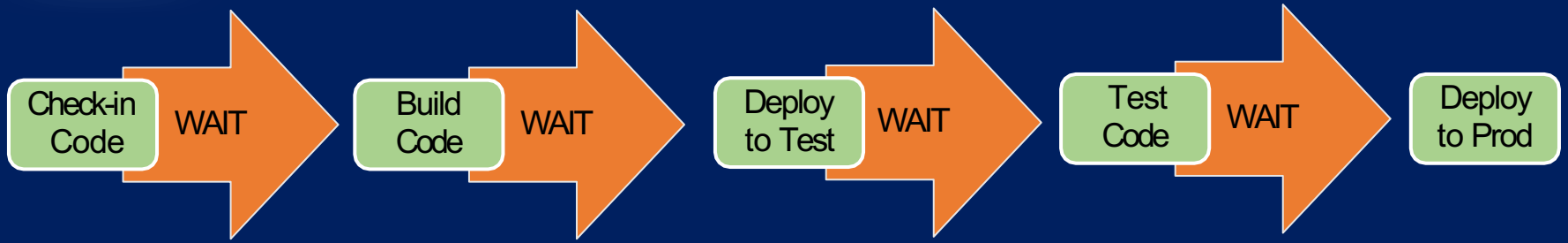


Hours/Days + Lot of Grief for Developer & Operations

Traditional Deployment

Developer

Operations



Hours/Days + Lot of Grief for Developer & Operations

Traditional Deployment

When are you gonna deploy my code?

Developer

When you stop breaking my servers

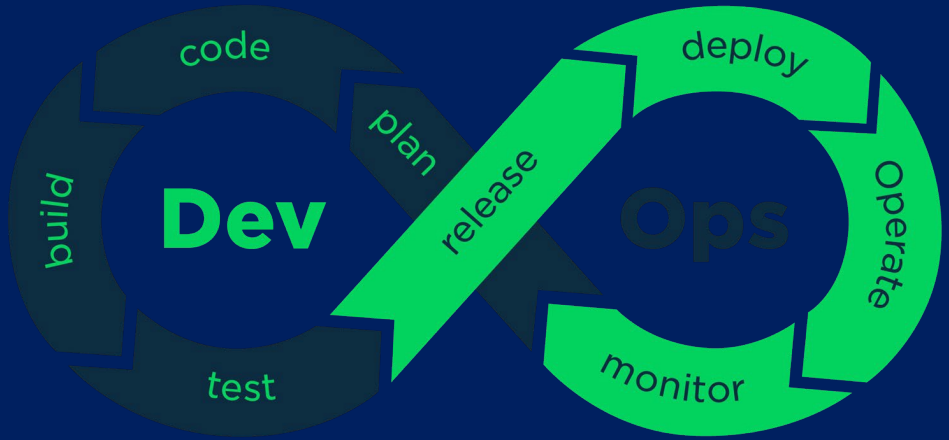
Operations

Check-in Code → WAIT → Build Code → WAIT → Deploy to Test → WAIT → Test Code → WAIT → Deploy to Prod


I just wanna do cool stuff
!
SYSTEM

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 “siloed”



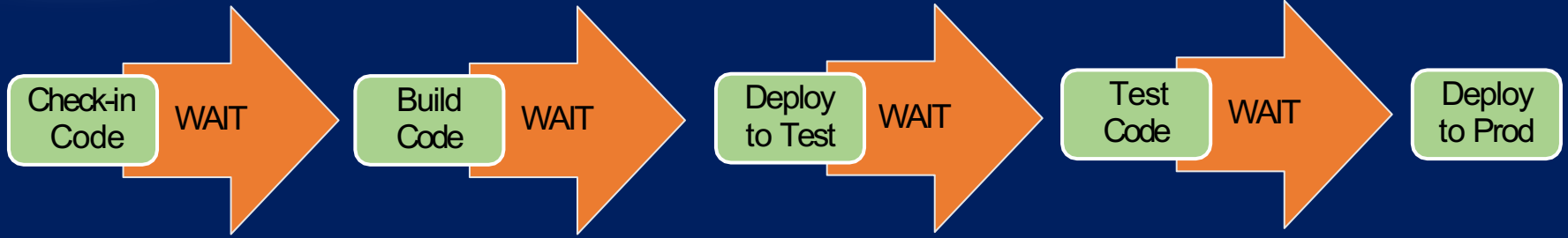
Traditional Deployment

When are you gonna deploy my code?

Developer

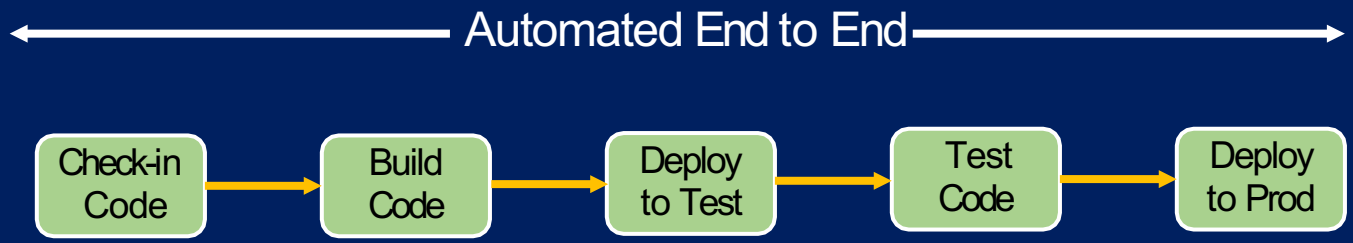
When you stop breaking my servers

Operations



DevOps

Same Team

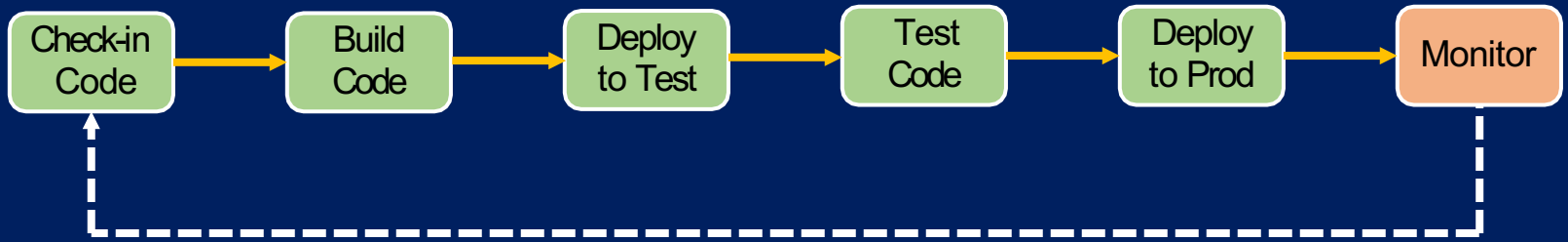


- Whole flow done in seconds!
- Easy to rollback in case of errors

DevOps

Same Team

Automated End to End



Rollback to Previous Version

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)

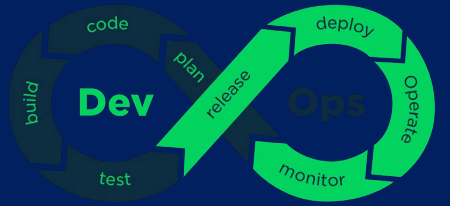
Cultural benefits

- 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

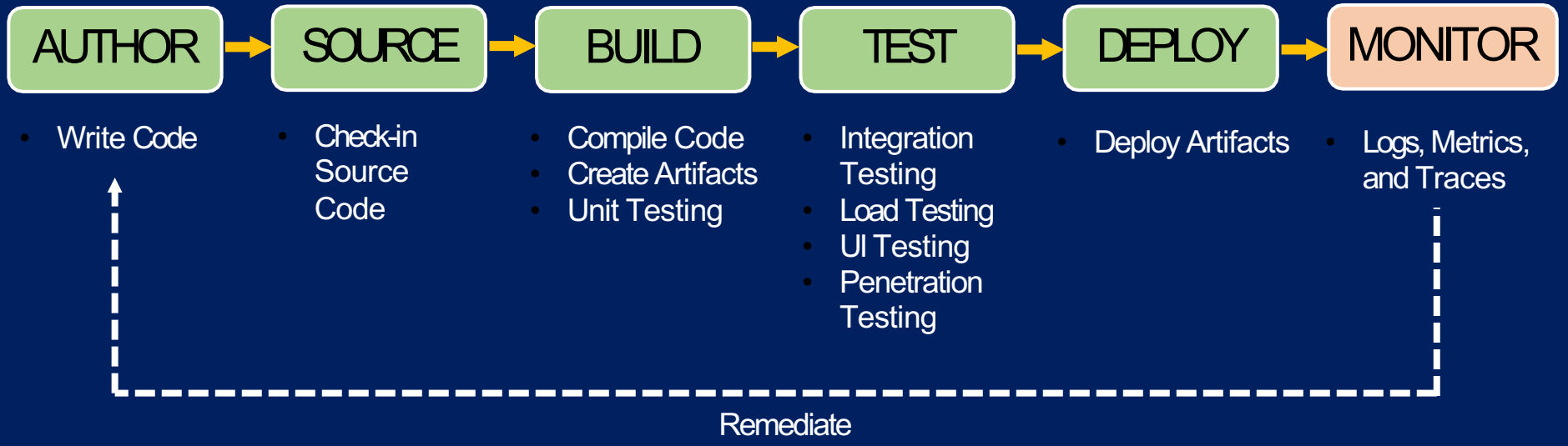
Same Team

Automated End to End

Check-in Code → Build Code → Deploy to Test → Test Code → Deploy to Prod → Monitor

Remediate

DevOps Phases



CI vs CD vs CD

AUTHOR → SOURCE → BUILD → TEST → DEPLOY → MONITOR

- Write Code
- Check-in Source Code
- Compile Code
- Create Artifacts
- Unit Testing
- Integration Testing
- Load Testing
- UI Testing
- Penetration Testing
- Deploy Artifacts
- Logs, Metrics, and Traces

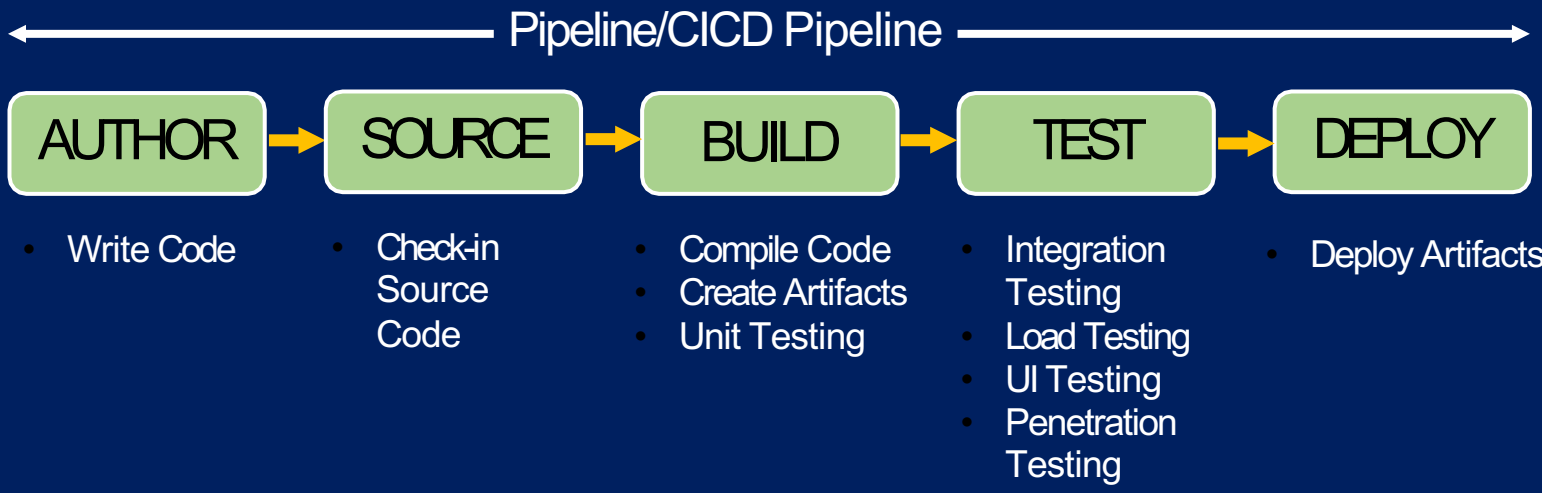
Continuous Integration (CI)

Continuous Delivery (CD)

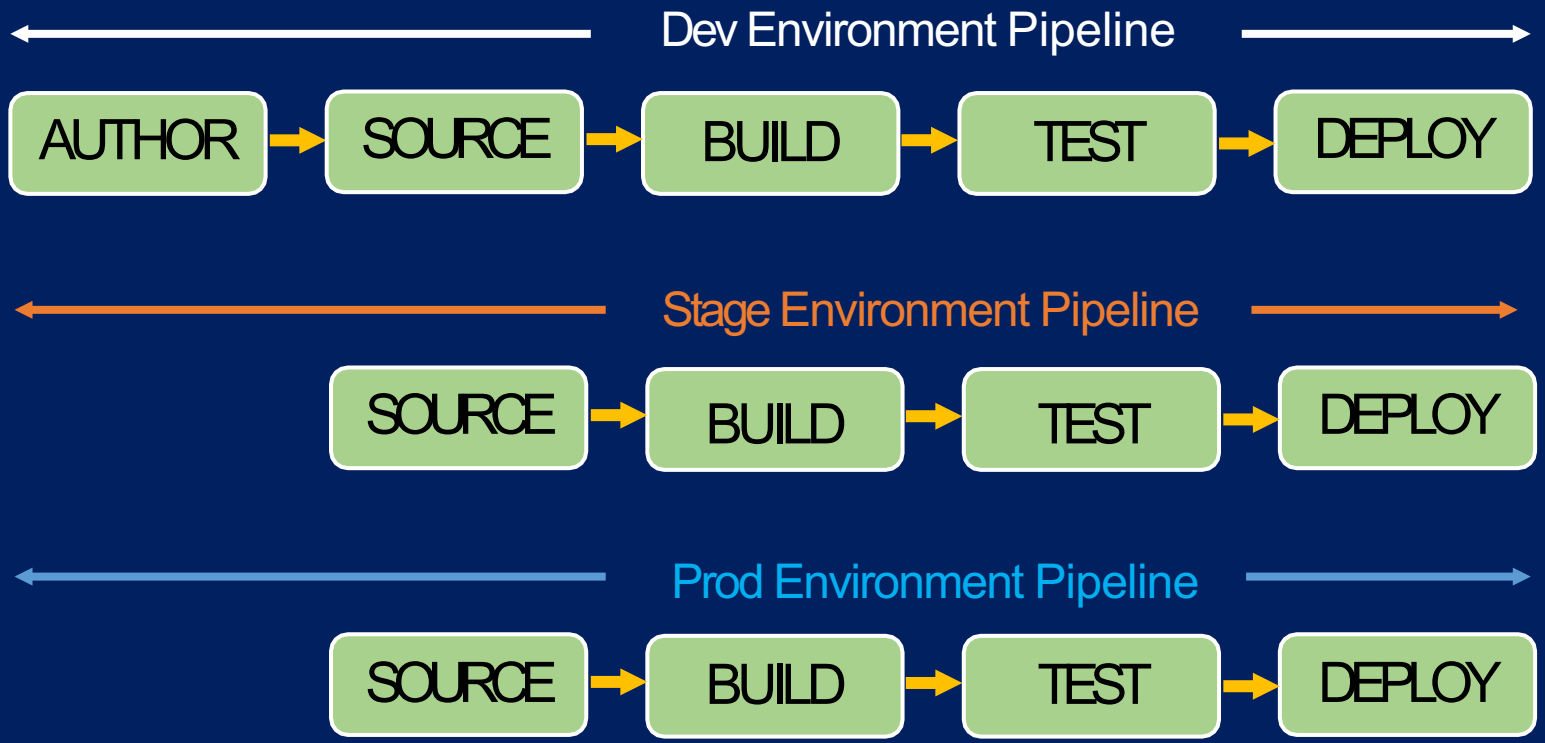
Manual Approval

Continuous Deployment (CD)

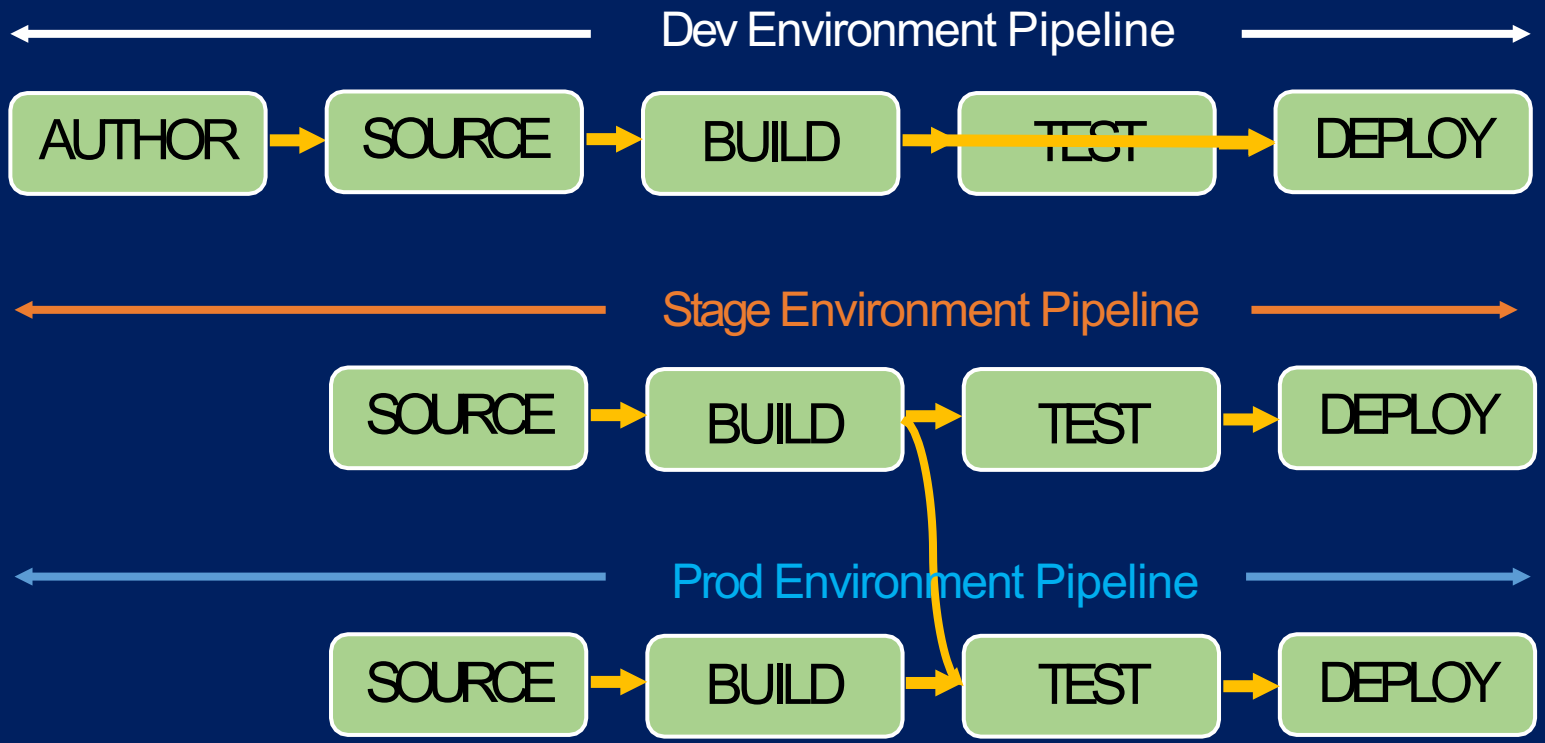
DevOps Phases



DevOps Pipelines



DevOps Pipelines



DevOps Tools

DevOps Phases

AUTHOR → **SOURCE** → **BUILD** → **TEST** → **DEPLOY** → **MONITOR**

- Write Code
- Check-in Source Code
- Compile Code
- Create Artifacts
- Unit Testing
- Integration Testing
- Load Testing
- UI Testing
- Penetration Testing
- Deploy Artifacts
- Logs, Metrics, and Traces

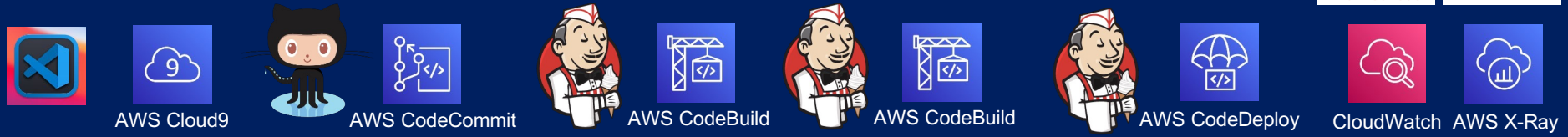
Continuous Integration (CI)

Continuous Delivery (CD)

Manual Approval

Continuous Deployment (CD)

DevOps Tools



AUTHOR → SOURCE → BUILD → TEST → DEPLOY → MONITOR

- Write Code
- Check-in Source Code
- Compile Code
- Create Artifacts
- Unit Testing
- Integration Testing
- Load Testing
- UI Testing
- Penetration Testing
- Deploy Artifacts
- Logs, Metrics, and Traces

Launching an AWS EC2 instance and Jenkins

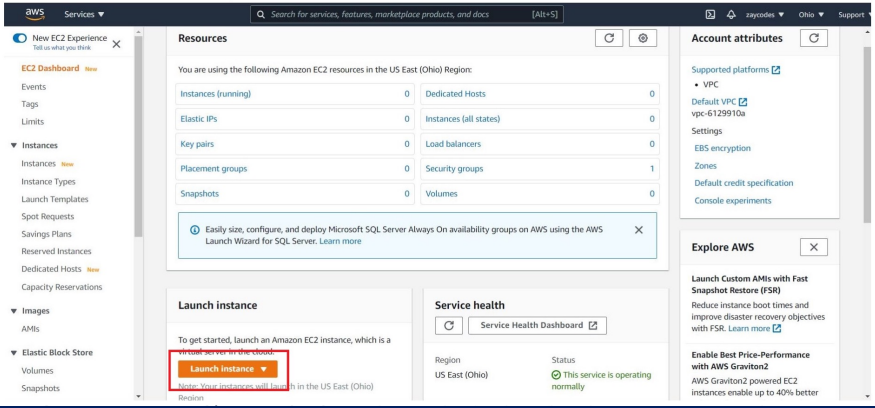
Launching an AWS EC2 instance and Jenkins on AWS

Launching an Amazon EC2 instance

Now that you have configured a key pair and security group, you can launch an EC2 instance.

To launch an EC2 instance:

1. Sign in to the the AWS Management Console.
2. Open the Amazon EC2 console by selecting EC2 under **Compute**.
3. From the Amazon EC2 dashboard, select **Launch Instance**.



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:lts-jdk11
```

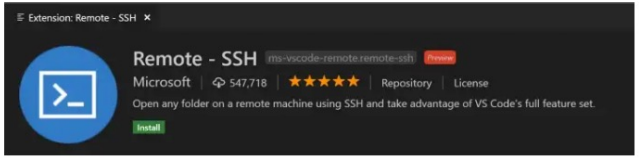
Using VSCode remotely on an EC2 instance

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.

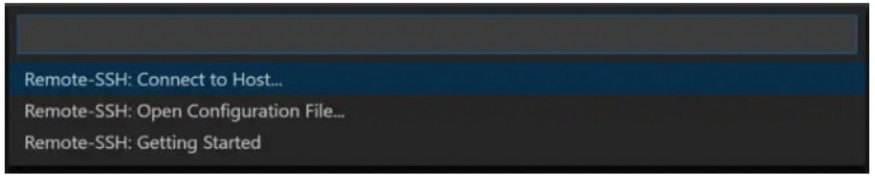
You can then search for the extension "**Remote-SSH**" in the VSCode marketplace.



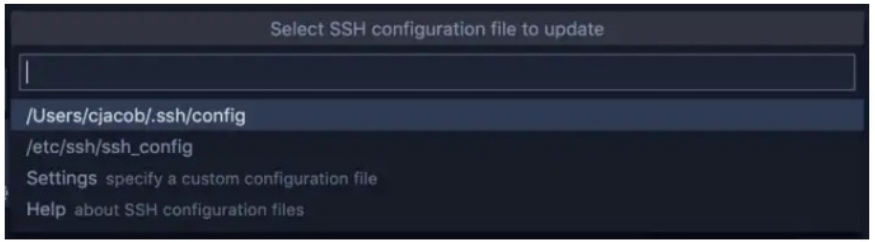
Once Installed, you should see a new Status bar item at the far left.

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
```

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

How to install Docker on Amazon Linux

https://www.cyberciti.biz/faq/how-to-install-docker-on-amazon-linux-2/

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 (1)
    stages {
        stage('Build') { (2)
            steps {
                // (3)
            }
        }
        stage('Test') { (4)
            steps {
                // (5)
            }
        }
        stage('Deploy') { (6)
            steps {
                // (7)
            }
        }
    }
}
```

1. Execute this Pipeline or any of its stages, on any available agent.
2. Defines the "Build" stage.
3. Perform some steps related to the "Build" stage.
4. Defines the "Test" stage.
5. Perform some steps related to the "Test" stage.
6. Defines the "Deploy" stage.
7. 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

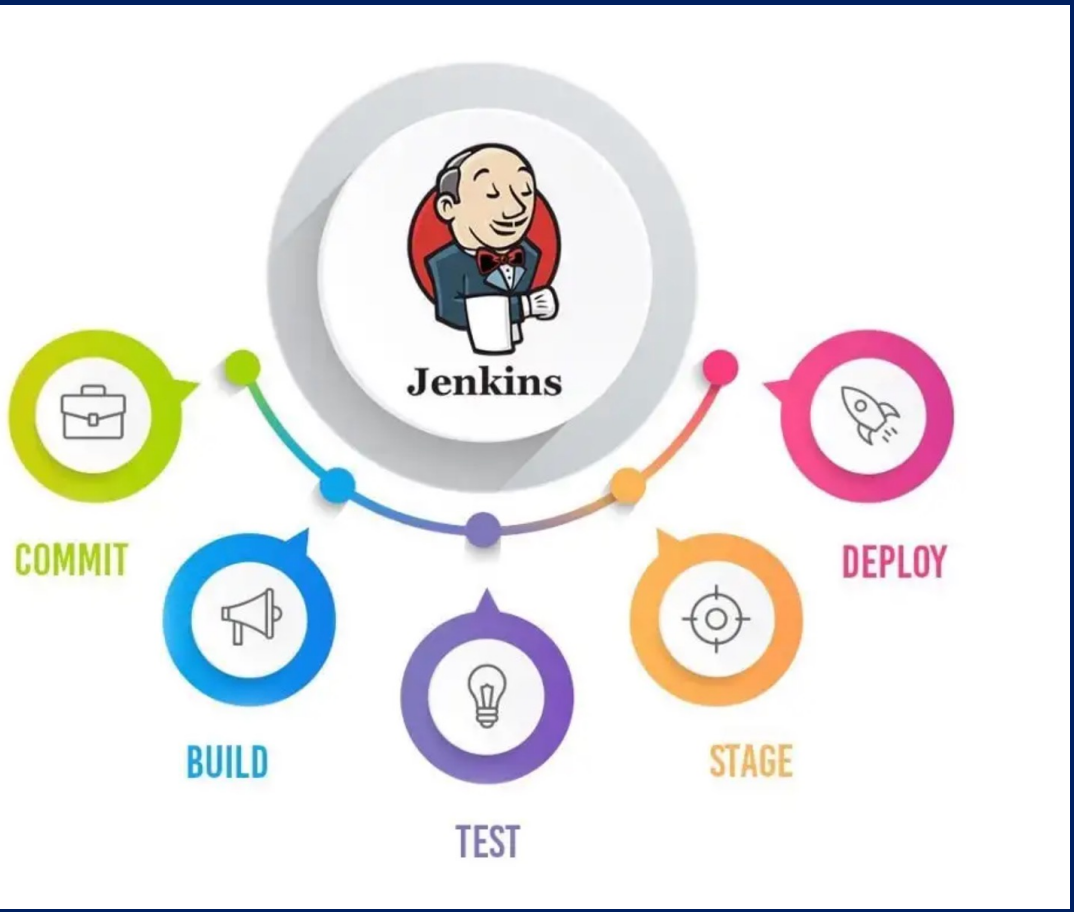
Jenkinsfile (Declarative Pipeline)

```
/* 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/

https://medium.com/@knoldus/setup-jenkins-in-docker-container-eb420f3a994

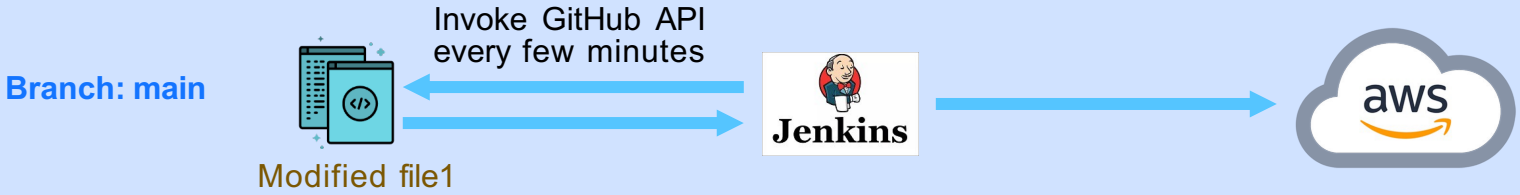
https://medium.com/tektutor/ci-cd-with-maven-github-docker-jenkins-aca28c252fec


Jenkins
COMMIT
BUILD
TEST
STAGE
DEPLOY

Branch: main

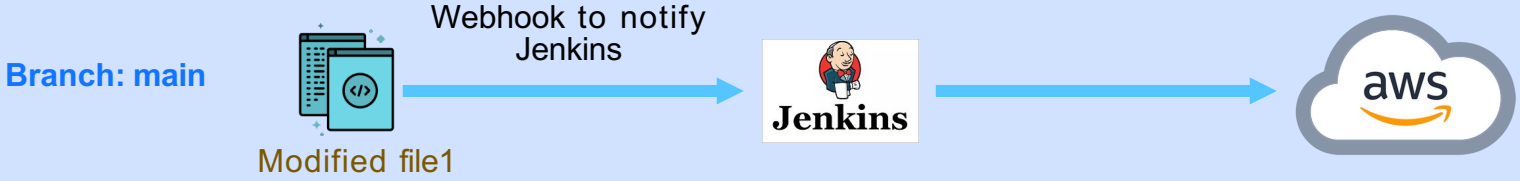
Modified file1

Calling API



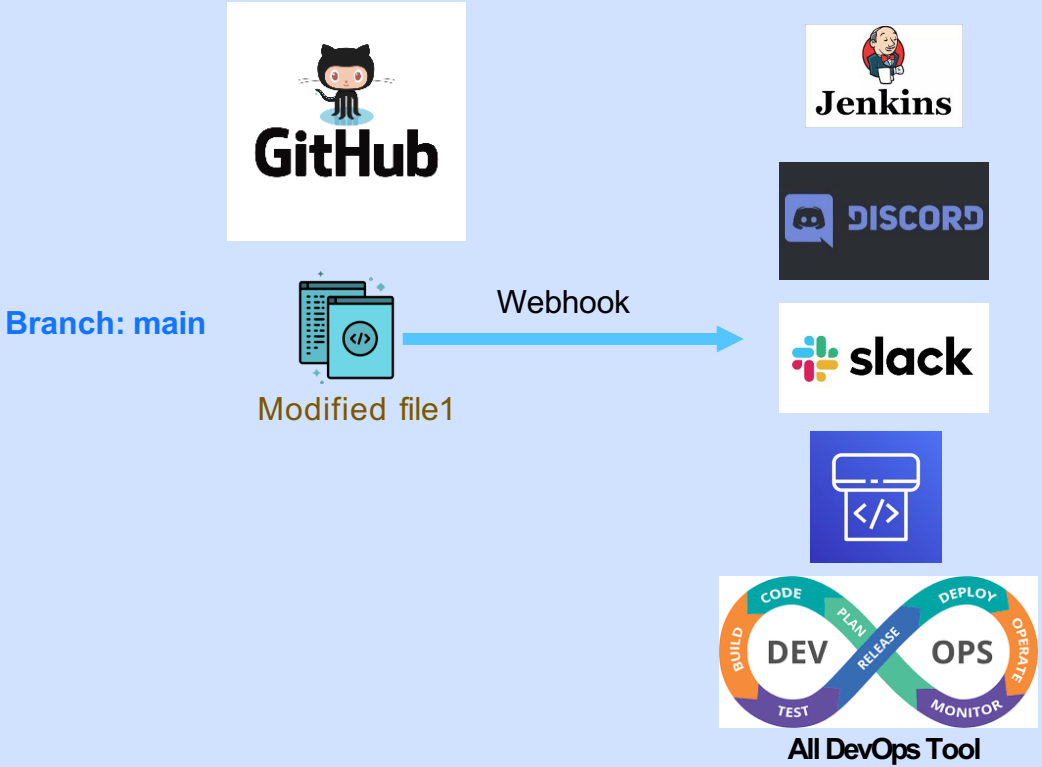
- Most of the times API will return stagnant data
- GitHub server will be bombarded
- Apps will exceed API limits

Webhook



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

Implementation



Jenkins & GitHub Integration with Selenium Python

https://www.youtube.com/watch?v=h2Abne2jIjM&t=248s