Lecture 2
DevOps

ECE 422: Reliable and Secure Systems Design

Instructor: An Ran Chen
Term: 2024 Winter

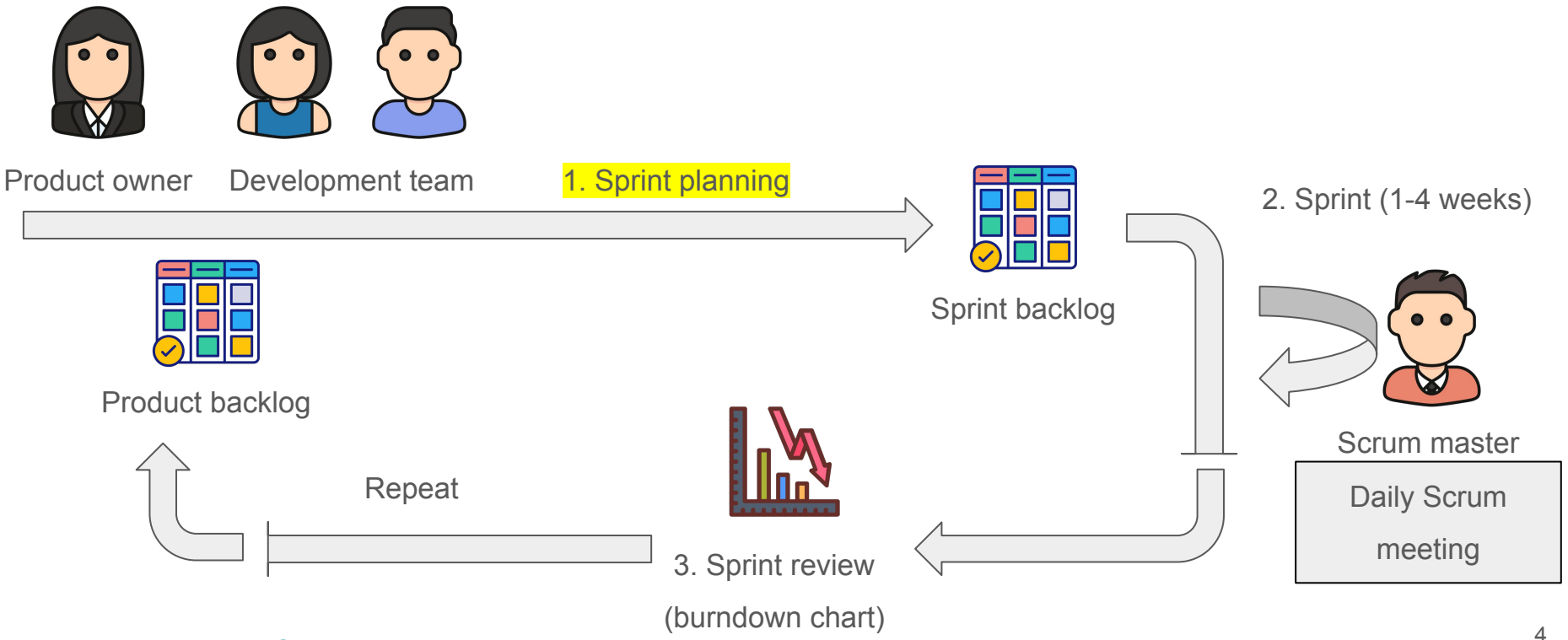
Schedule for today

- Key concepts from last class
- DevOps
- CI/CD
- Docker container
- TODOs for upcoming classes

Agile methodology

- Suggests an iterative approach to deliver a project, different from traditional approaches (e.g., waterfall model)
  - Focuses on continuous releases
  - Flexibility to adjust and iterate during the development process
- Cornerstone of DevOps practices
  - By fostering collaboration and feedback (both with the customer and within the team)
  - Deliver earlier = detect faults earlier, less expensive to fix (developing software as building house, software faults as cracks in house’s foundation)

Scrum, an agile project management framework



Icon created by Jesus Chavarria and Vectors Tank

User stories

User stories: a software feature written from the end user's perspective.

- Written in non-technical language to provide context
- Why? To describe what value a software feature provides to the customer
- An user story should explain: persona + need + purpose

As a [persona], I want [need], so that [purpose].

- Persona: Who is the end user?
- Need: What is the goal of the end user?
- Purpose: What problem does the end user look to solve?

Example: As a sales director, I want a sales report, so that I can monitor the sales progress.

Schedule for today

- Key concepts from last class
- DevOps
- CI/CD
- Docker container
- TODOs for upcoming classes

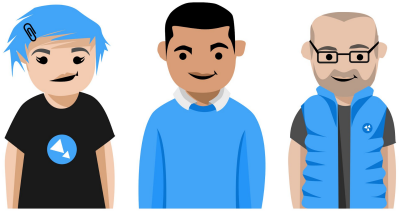
DevOps

DevOps is a partnership between software developers (Dev) and IT operators (Ops).

- Speeds up deployments
- Improves communication and collaboration
- Ensures quality and reliability of the system

Note: in the next class, we will see that DevOps is not a perfect solution if we are prioritizing on the reliability of the system.

Traditional software development process



Developers

Responsibilities

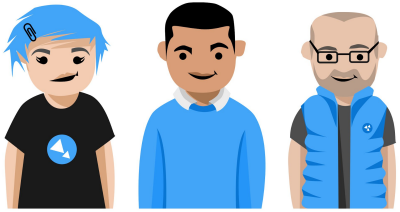
- Design
- Code
- Test
- Release engineering
- Improvements

Operators

Responsibilities

- Deploy
- Monitor
- Troubleshoot
- Anomaly detection
- Quality assurance
- Configuration tuning

Traditional software development process



Developers

Responsible for development of the system

1. looking to implement more features
2. want to release faster to the customers
3. work on the development and testing environments (no idea on how the system is deployed)

Operators

Responsible for the operations of the system

1. keeping the system stable and safe
2. ensure that the system is well-tested before the release
3. work on the production environment (no idea how the system is implemented and tested)

Challenges in traditional software development

- Poor communication
  - No collaboration between developer and operations teams
    - Developers implement the system
    - Operators deploy and operate the system
    - However, the operation teams are responsible for troubleshooting the problems in deployment while they lack the implementation information
  - Consequence: slow down the releases
- Isolated expertises
  - Because of the knowledge gap, both teams struggle in resolving problems
    - Developers have limited insights into operations
    - Operations staff have limited capacity in resolving anomalies

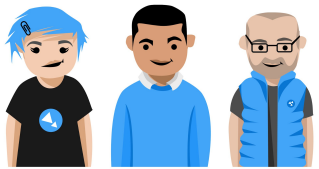
Challenges in traditional software development

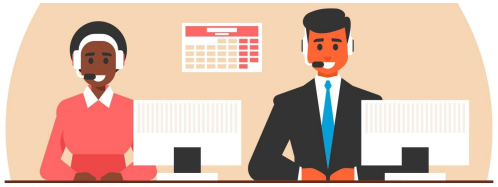
- Conflicting goals
  - New features vs stability
    - Developers look to push new features quickly to the customer
    - Operators want to keep the system stable and safe
    - Operators want to ensure there are as little change to the infrastructure of the system as possible
- Different cultures
  - Common goal of being as productive as possible, but
    - Developers want to improve the system as much as possible
    - Operators want to limit the changes to reduce uncertainty

DevOps

DevOps promotes a collaborative culture, where development and operations teams work together to share responsibilities for software correctness and reliability.

- It focuses on reducing time to market for new features while ensuring high quality.





Development and operations teams work together under the **DevOps workflow** to shorten the development life cycle, provide continuous delivery while ensuring high quality.

DevOps

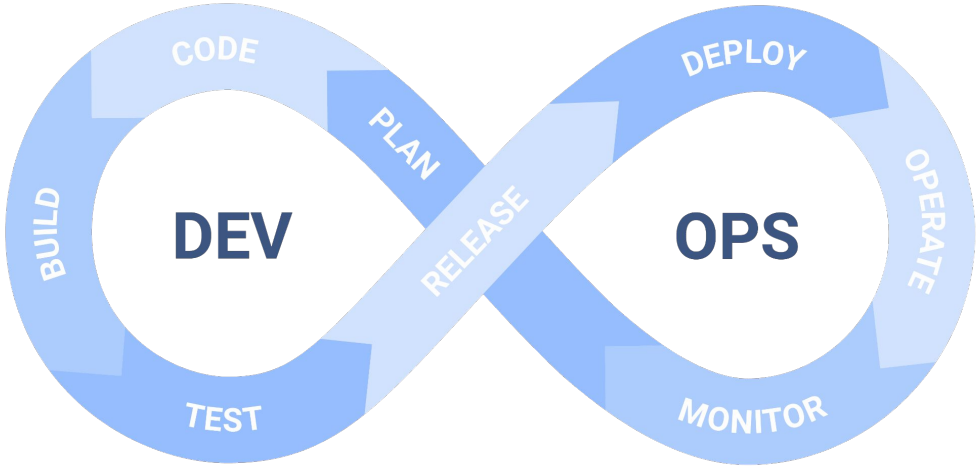
DevOps has three main components:

- Workflow
- Automation
- Tool supports

DevOps workflow

DevOps workflow is continuously looping through a set of activities between development and operations.

- involves continuous development, testing, monitoring, feedback, and deployment processes.



DevOps workflow

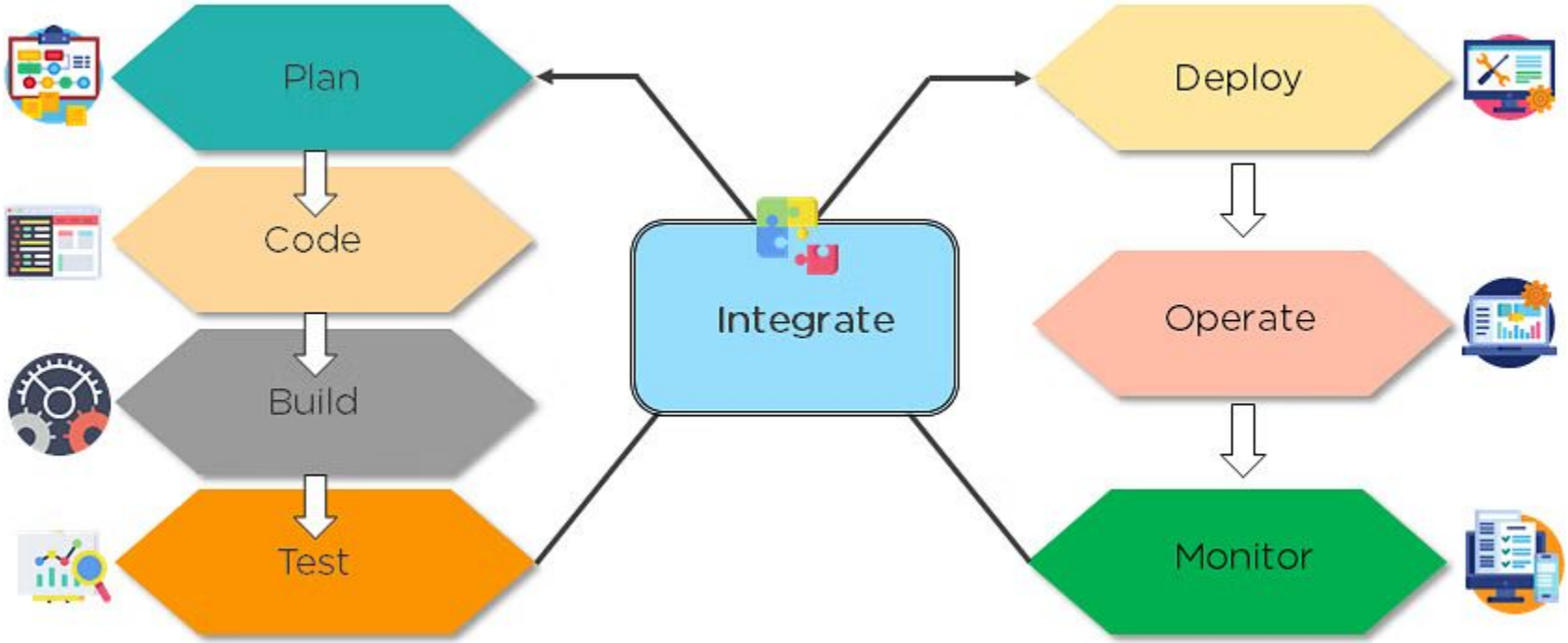
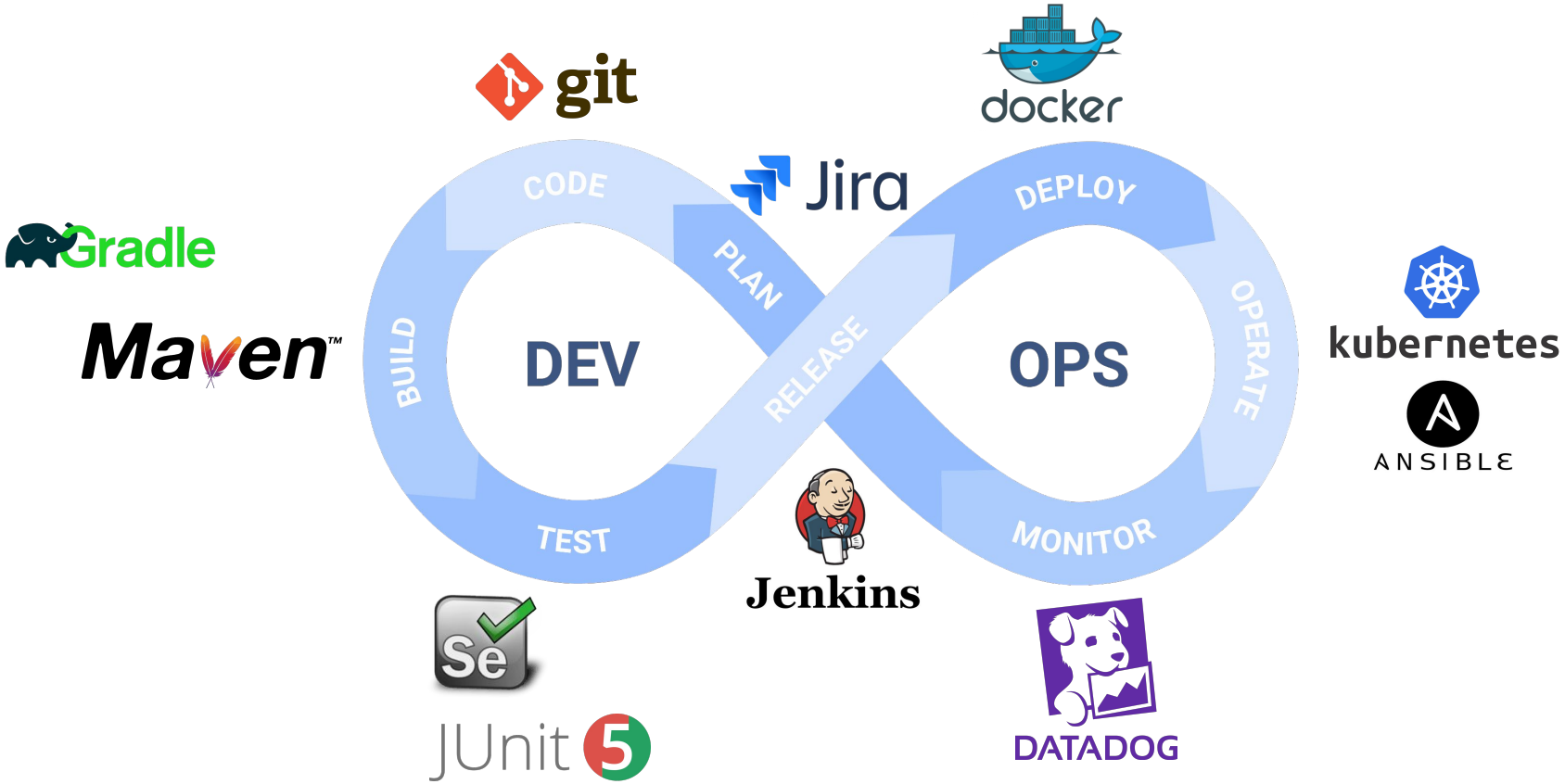


Image by Simplilearn

DevOps workflow



DevOps

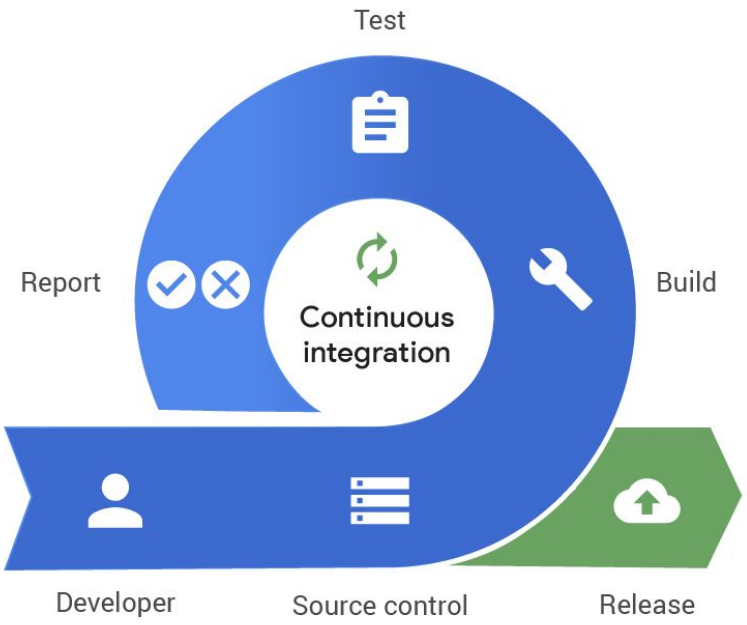
DevOps has three main components:

- Workflow
- Automation
  - DevOps uses automation to performs tasks which can reduce human assistance and facilitate feedback loops between operations and development teams.
  - E.g., CI/CD
- Tool supports

CI/CD

- CI stands for Continuous Integration
- CD can be Continuous Delivery and/or Continuous Deployment
- CI/CD aims to streamline and speed up the development and delivery processes
    - CI/CD automates the development process from an initial code change to its deployment to the production environment.

Continuous Integration (CI)



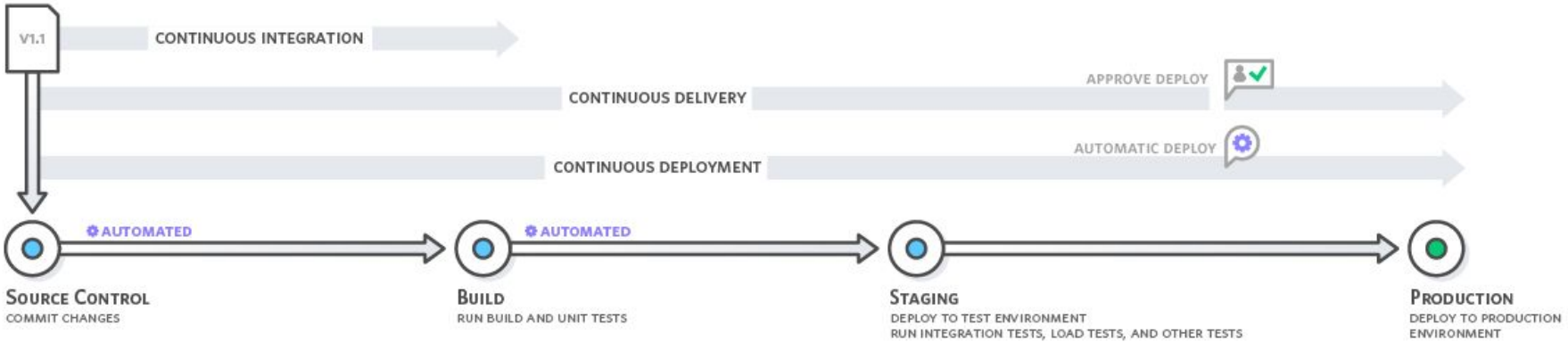
Continuous integration is the practice of automatically and frequently integrating code changes into a shared source code repository.

- Why? To detect integration issues early
- Developers frequently merge their code changes into a central repository
- Each commit triggers an automated workflow that builds and tests the system

Continuous Delivery (CD)

Continuous delivery is the practice of automatically releasing new code changes through automated testing process into the production environment with manual approval.

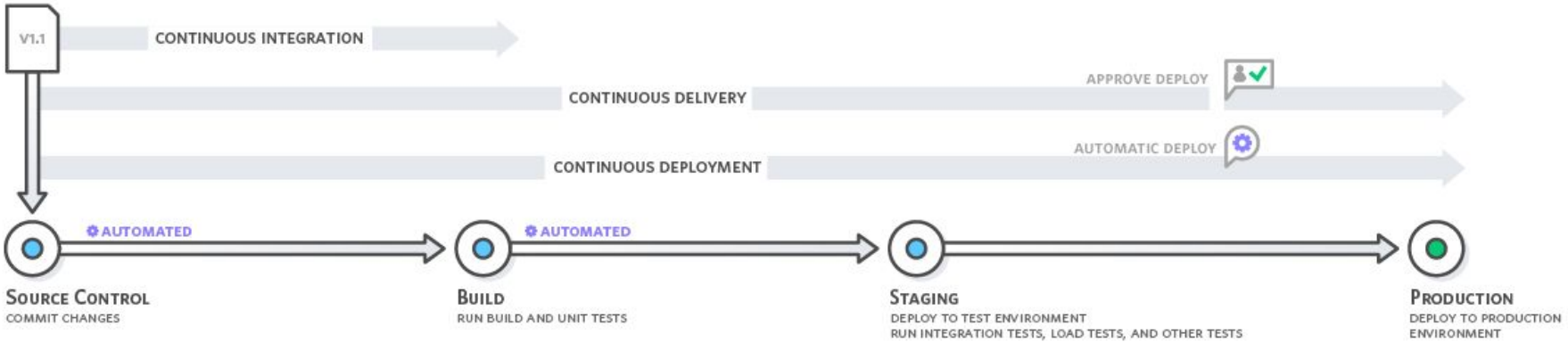
- Why? To deliver updates to customers faster and more frequently
- Every code change is built and tested, and then pushed to staging environment.
- Once the testing finishes on staging, the final decision to deploy to a live production environment is triggered manually.



Continuous Deployment (CD)

Continuous deployment is the practice of automatically releasing new code changes through automated testing process directly into the production environment.

- Why? To reduce manual processes
- Every code change is built and tested, and then pushed to staging environment.
- Once the testing finishes on staging, the changes are automatically deployed to the live production environment.



Continuous delivery vs continuous deployment

1. Ensure that code can always be safely deployed on the production servers

   a) continuous delivery, b) continuous deployment, c) both

2. Make software releases faster and more robust

   a) continuous delivery, b) continuous deployment, c) both

3. Ensure that the application works as expected

   a) continuous delivery, b) continuous deployment, c) both

Continuous delivery automates the entire process up to the point of release readiness, but the actual release to production is triggered manually.

Continuous deployment automates the entire process, including the release to production, eliminating the need for manual intervention in deploying changes.

Continuous delivery vs continuous deployment

1. Ensure that code can always be safely deployed on the production servers

   a) continuous delivery, b) continuous deployment, c) both

2. Make software releases faster and more robust

   a) continuous delivery, b) continuous deployment, c) both

3. Ensure that the application works as expected

   a) continuous delivery, b) continuous deployment, c) both

Continuous delivery automates the entire process up to the point of release readiness, but the actual release to production is triggered manually.

Continuous deployment automates the entire process, including the release to production, eliminating the need for manual intervention in deploying changes.

Continuous delivery vs continuous deployment

4. Deliver every change through rigorous automated testing

   a) continuous delivery, b) continuous deployment, c) both

5. Every change that passes the automated tests is deployed to production automatically

   a) continuous delivery, b) continuous deployment, c) both

6. Deployed with no explicit approval but require a culture of monitoring

   a) continuous delivery, b) continuous deployment, c) both

Continuous delivery automates the entire process up to the point of release readiness, but the actual release to production is triggered manually.

Continuous deployment automates the entire process, including the release to production, eliminating the need for manual intervention in deploying changes.

Continuous delivery vs continuous deployment

4. Deliver every change through rigorous automated testing

   a) continuous delivery, b) continuous deployment, c) both

5. Every change that passes the automated tests is deployed to production automatically

   a) continuous delivery, b) continuous deployment, c) both

6. Deployed with no explicit approval but require a culture of monitoring

   a) continuous delivery, b) continuous deployment, c) both

Continuous delivery automates the entire process up to the point of release readiness, but the actual release to production is triggered manually.

Continuous deployment automates the entire process, including the release to production, eliminating the need for manual intervention in deploying changes.

DevOps

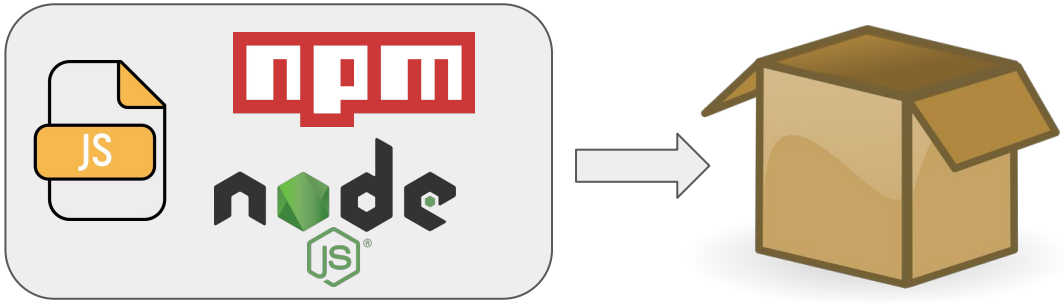
DevOps has three main components:

- Workflow
- Automation
- Tool supports
  - DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity (story points)
  - E.g., containerization

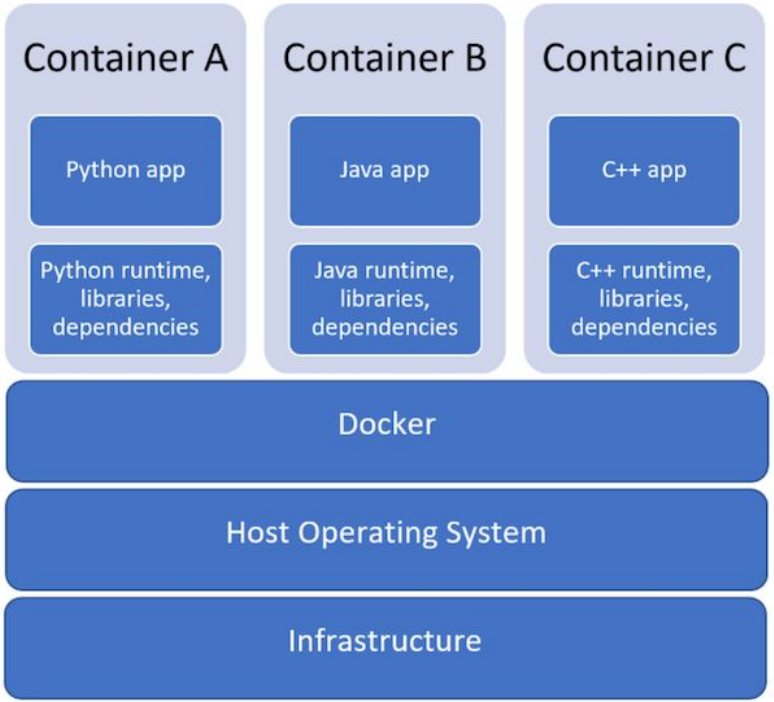
Containerization

A process that virtualizes resources for software application

- Packages an application with all the files and libraries it needs so it can run on any infrastructure
    - E.g., the dependencies, configuration, system tools and runtime become one standardized unit, *the container*
- Helps to build, test, and deploy applications easier and quicker



Container



Example

A Python container include:

- Python application
- Python runtime
- Application dependencies

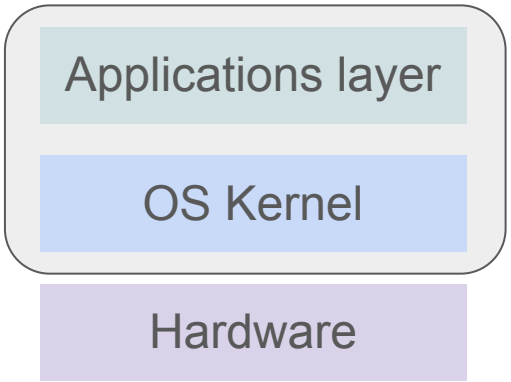
Image form [Visual studio code]

Docker container

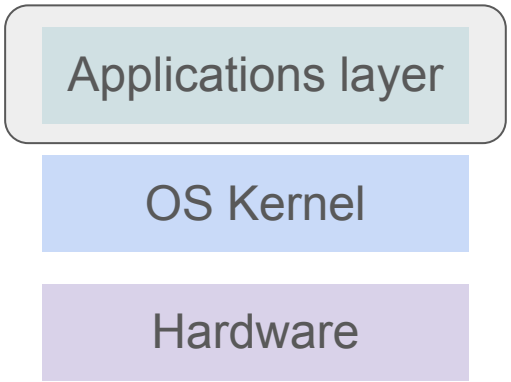
Docker packages software into standardized units (also known as containers) that include everything the software needs to run.

It works in a similar way as a virtual machine

VM: virtualization of the OS Kernel + Application layer



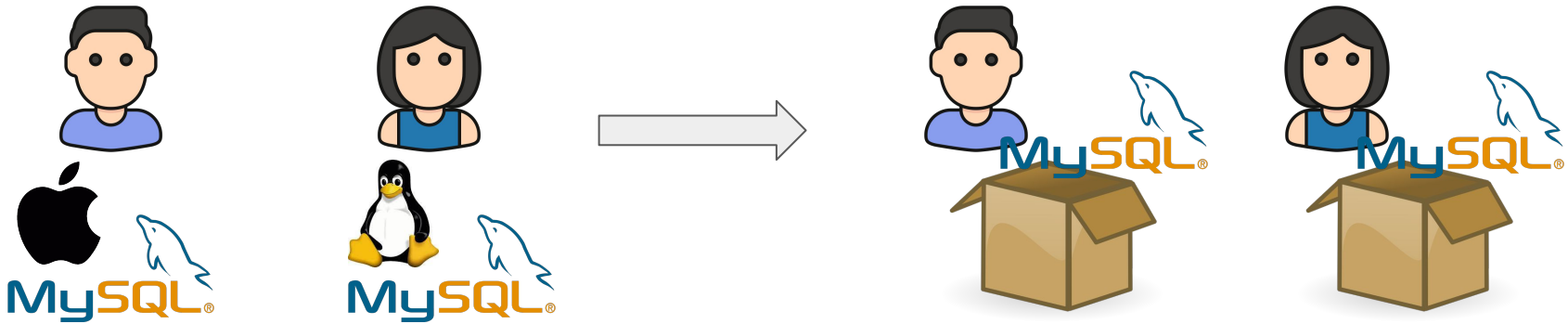
Docker: virtualization of the Application layer only



Docker container

Containers help to

- Standardize services on all local development environments
- Contain all services, dependencies and configurations in one place
- Isolate dependencies from the local machine



Docker container on security

Docker is also effective against malicious third-parties dependencies as it has its own filesystem.

- Protect against typosquatting attacks
  - Typosquatting attacks create code packages that include a typo to the packages' names (e.g., atlas-client rather than atlas_client)
  - Hoping the victims install the incorrect version to deliver the malware
- Container only have accesses to its own files
- Avoid ransomware attack as we run Node.js inside the container

Docker container in DevOps

Development team can package everything into docker container, and send it to the operations team for deployment.

- Avoid installations and configurations on the server’s operating system
  - Resolve dependency version conflicts on the server
- Mitigate the possibility of miscommunication
  - No manual configurations needed on the server

Dockerfile

```
FROM node:21-alpine

WORKDIR /app

COPY src /app/
COPY package.json /app/

RUN npm install \
      && npm run build

CMD [ "serve", "-s", "build" ]
```

- FROM instruction creates a base image
  - Starting point for the image
  - An empty first layer, e.g., node:21-alpine
  - List of images available on Docker Hub

Dockerfile

```
FROM node:21-alpine

WORKDIR /app

COPY src /app/
COPY package.json /app/

RUN npm install \
      && npm run build

CMD [ "serve", "-s", "build" ]
```

- WORKDIR instructions changes the working directory
  - Set /app as the main working directory
- COPY instructions copies local files
  - The src directory and package.json file are added into the container at path /app
- RUN instruction installs node packages and build the app
  - Install the dependencies inside the container

Dockerfile

```
FROM node:21-alpine

WORKDIR /app

COPY src /app/
COPY package.json /app/

RUN npm install \
      && npm run build

CMD [ "serve", "-s", "build" ]
```

- CMD instruction to run the app where the container is ran
  - Last instruction in Dockerfile
  - There can only be one CMD instruction in a Dockerfile

More information on the instructions: Docker docs

Benefits of Docker containers

- Modularity
  - Part of an application can be taken down for updates
- Layer and image version control
  - Each Docker image contains a series of layers
  - Full control over container images
- Rollback
  - Each Docker image contains a series of layers
  - Roll back the image to the previous version
- Rapid Deployment
  - Portable artifact
  - Easy to share and distribute

DevOps

DevOps is a partnership between software developers (Dev) and IT operators (Ops).

- Improves communication and collaboration
  - Developers and operators share responsibilities and combine work.
- Speeds up deployments
  - Reduce time between committing a change and shipping that change into production environment.
- Ensures quality and reliability of the system
  - Practices like continuous integration and continuous delivery ensure changes are functional and safe.

Project deliverable

Project 1: Reliability Auto-Scaling Deliverable (3-5 pages)

- Due Wednesday, January 24

Design

- Class diagram

Tools and technologies

- What technologies you plan to use? Why?

User stories

- Two user stories (persona + need + purpose)
- Each user story should be broken down into sub-tasks

Planning

- Timeline of the subtasks

TODOs

- In Slack #teams channel: post your team information before Friday 23:59 pm
  - Email me or send a private message on Slack if you need help finding a team. You are also welcome to send a *short* message in the #teams channel to find a team yourself.
- Check out the Cybera instructions shared by the TAs on Slack
- Check out the Docker documentations on eClass