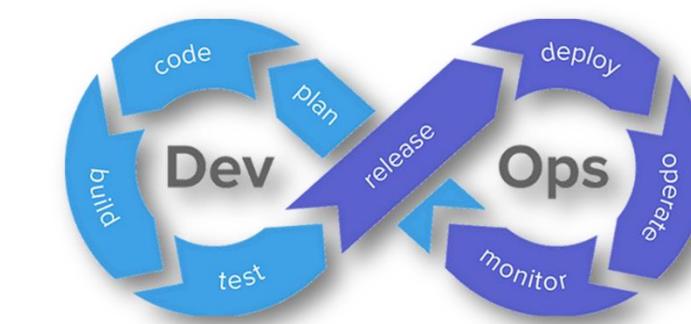


Practical DevOps Bootcamp for All





Puppet for the Absolute Beginners – Hands-On



SaltStack for the Absolute Beginners – Hands-On



Infrastructure Automation with OpenTofu – Hands-On



AI Ecosystem for the Absolute Beginners – Hands-On



Podman for the Absolute Beginners – Hands-On



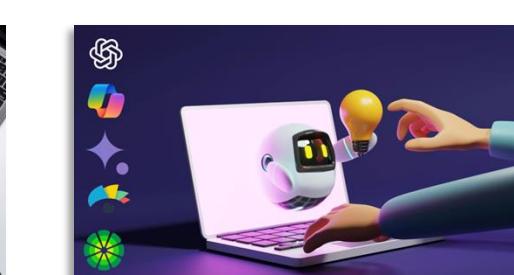
Generative AI Essentials - Practical Use Cases



Mastering Docker Essentials - Hands-on



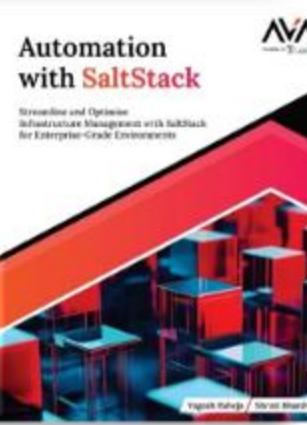
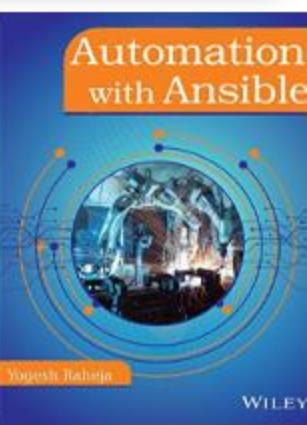
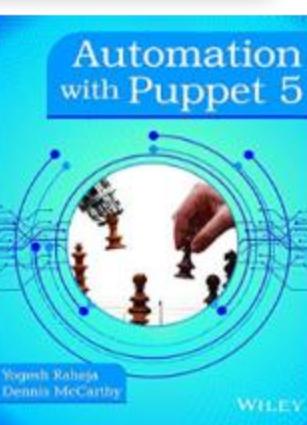
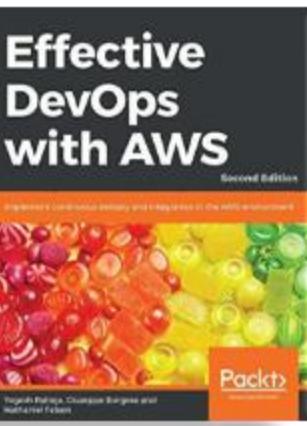
Unlocking Python for the Absolute Beginners



Mastering Prompt Engineering for GenAI



Practical Kubernetes - Beyond CKA and CKAD

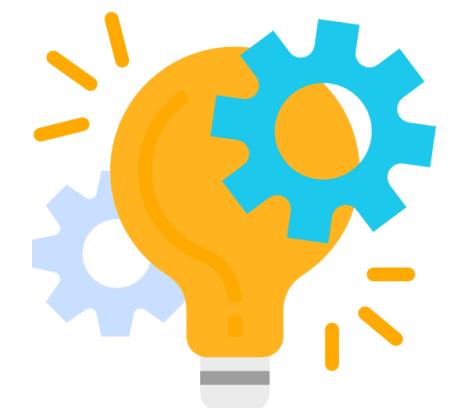


Thinknyx
You Trust, We Deliver!

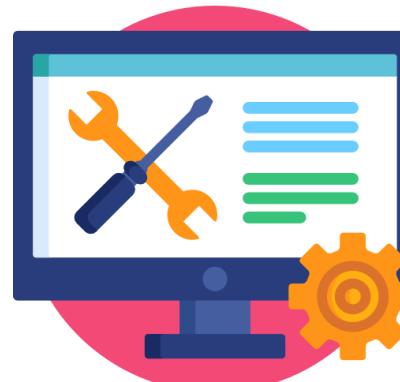




One-Stop Destination



Concepts



Tools

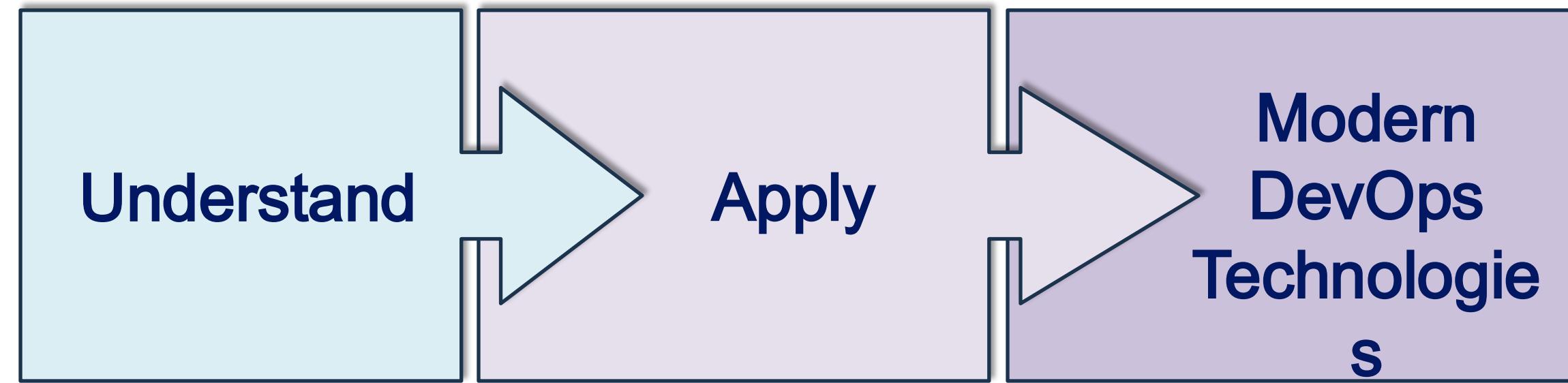


Practical Skills



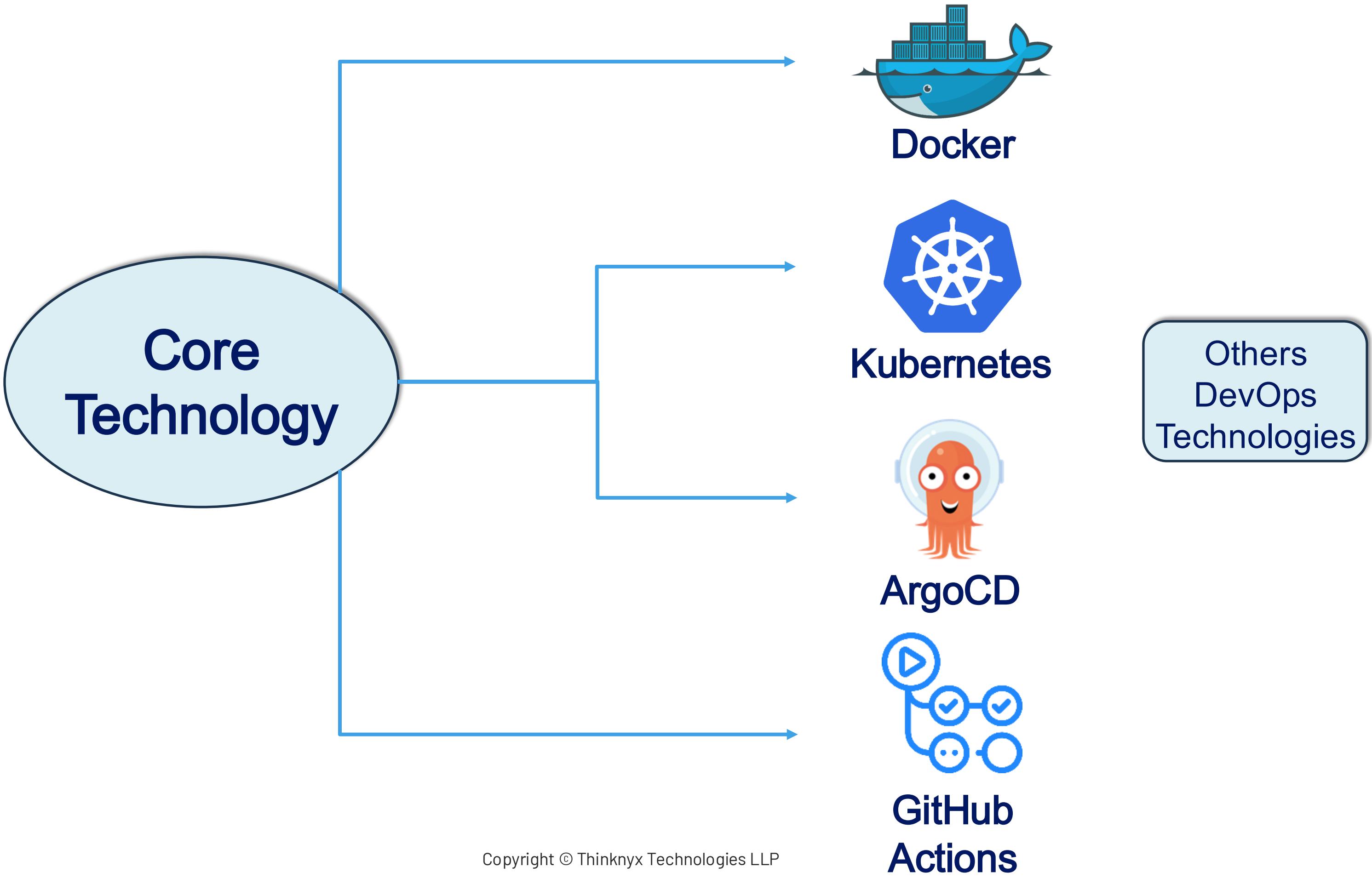
How Will This Course Work?

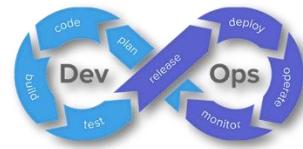
□ Goal





How Will This Course Work?





How Will This Course Work?



Practice Lectures

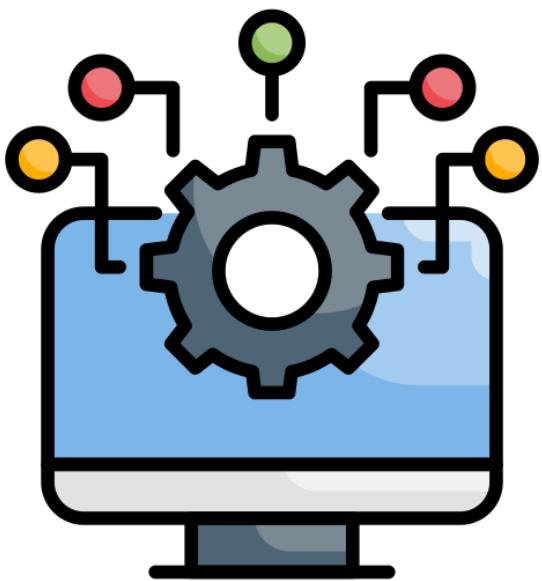


Hands-On Demonstrations

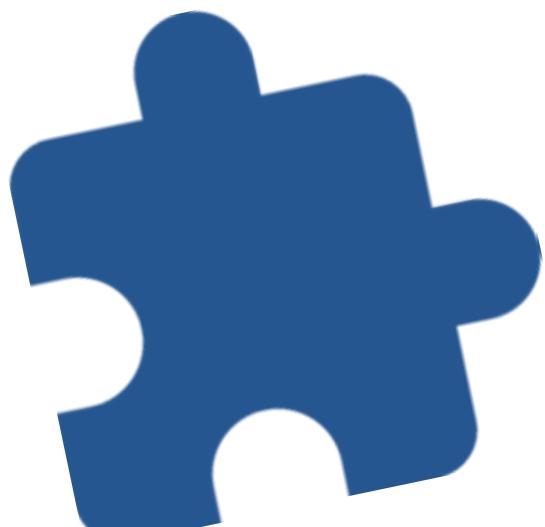


How Will This Course Work?

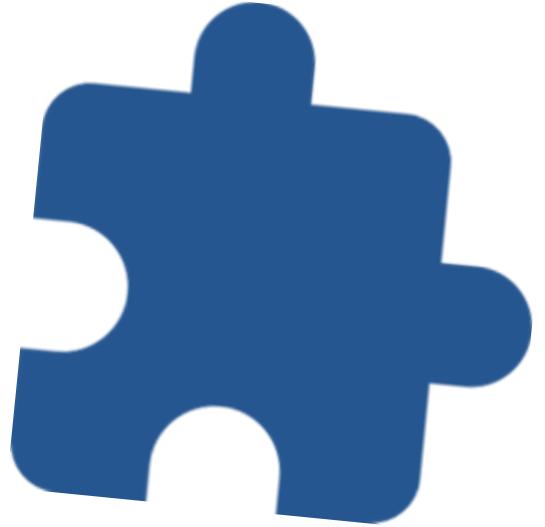
- At the end of every section



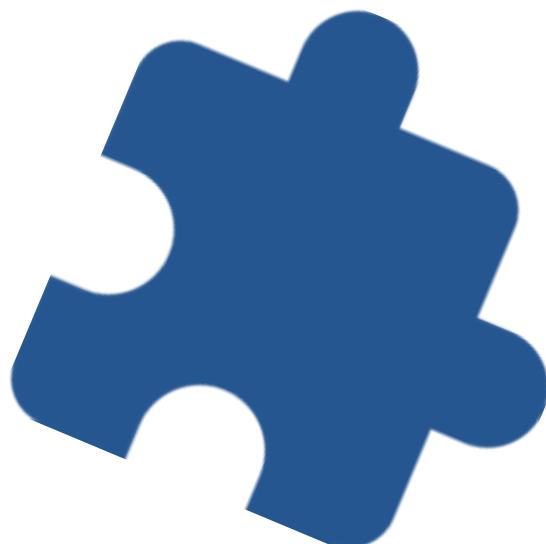
Project Integration Module



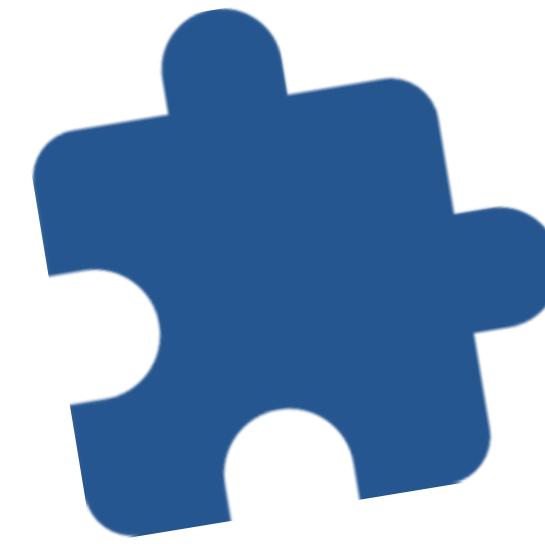
Technology 1



Technology 2



Technology 3

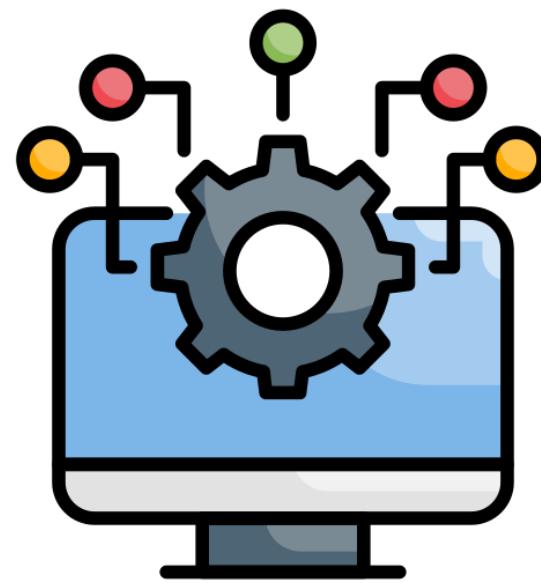


Technology 4

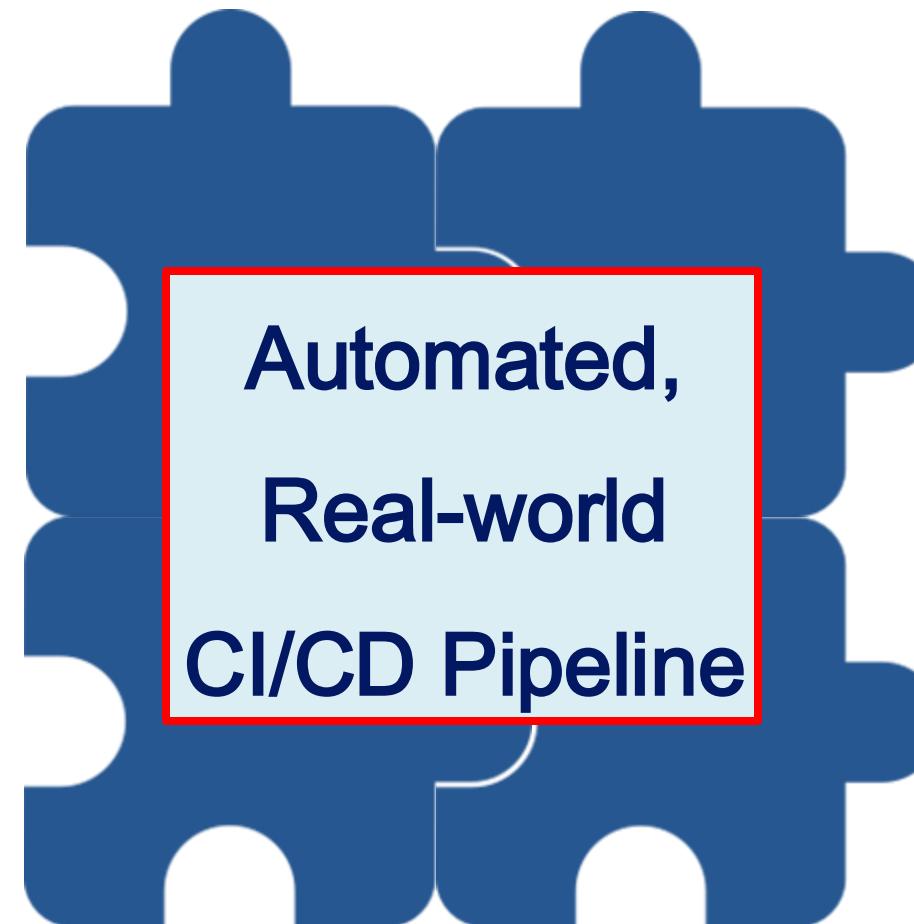


How Will This Course Work?

- ❑ At the end of every section



Project Integration Module



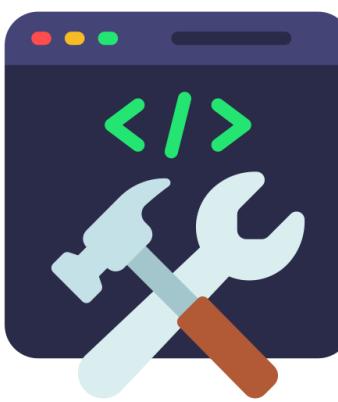
- ✓ Understand DevOps tools and how they work together in real environments, on real projects



How Will This Course Work?



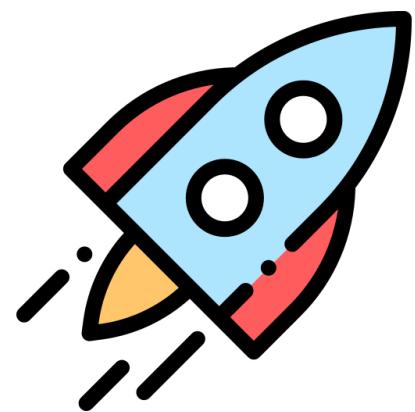
Learn



Build



Experiment

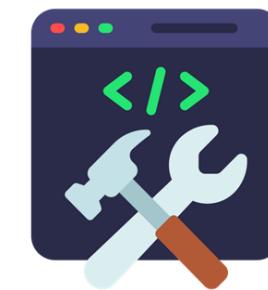
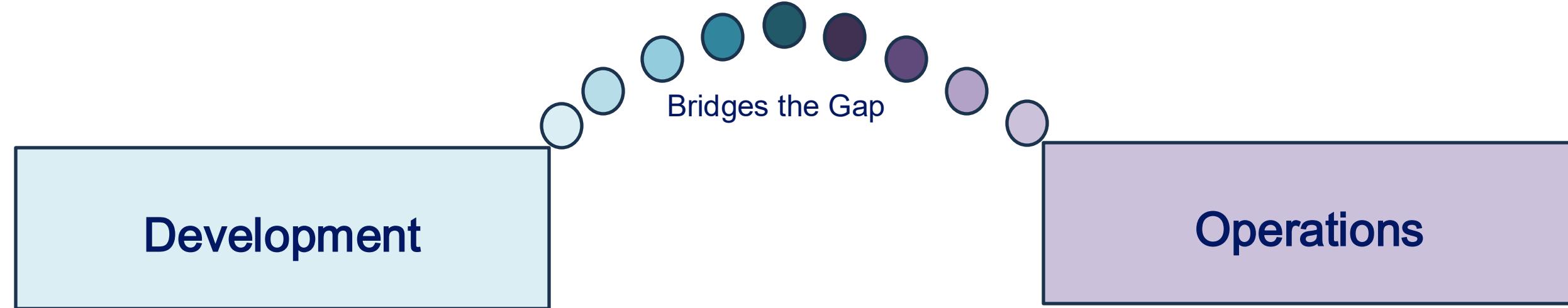


Action

Course Goal



Course Goal



Tools



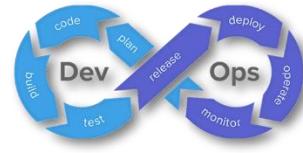
Collaboration



Automation



Deliver



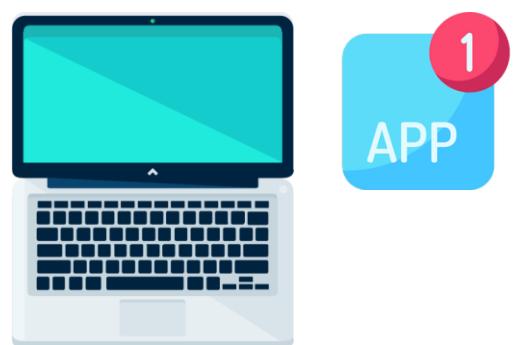
Course Goal



- ✓ Book App – Easy browsing and ordering
- ✓ Agile Delivery – Fast releases without breakage
- ✓ Scalable System – Reliable, monitored, and growth-ready

DevOps
Engineer

Challenge





Course Goal



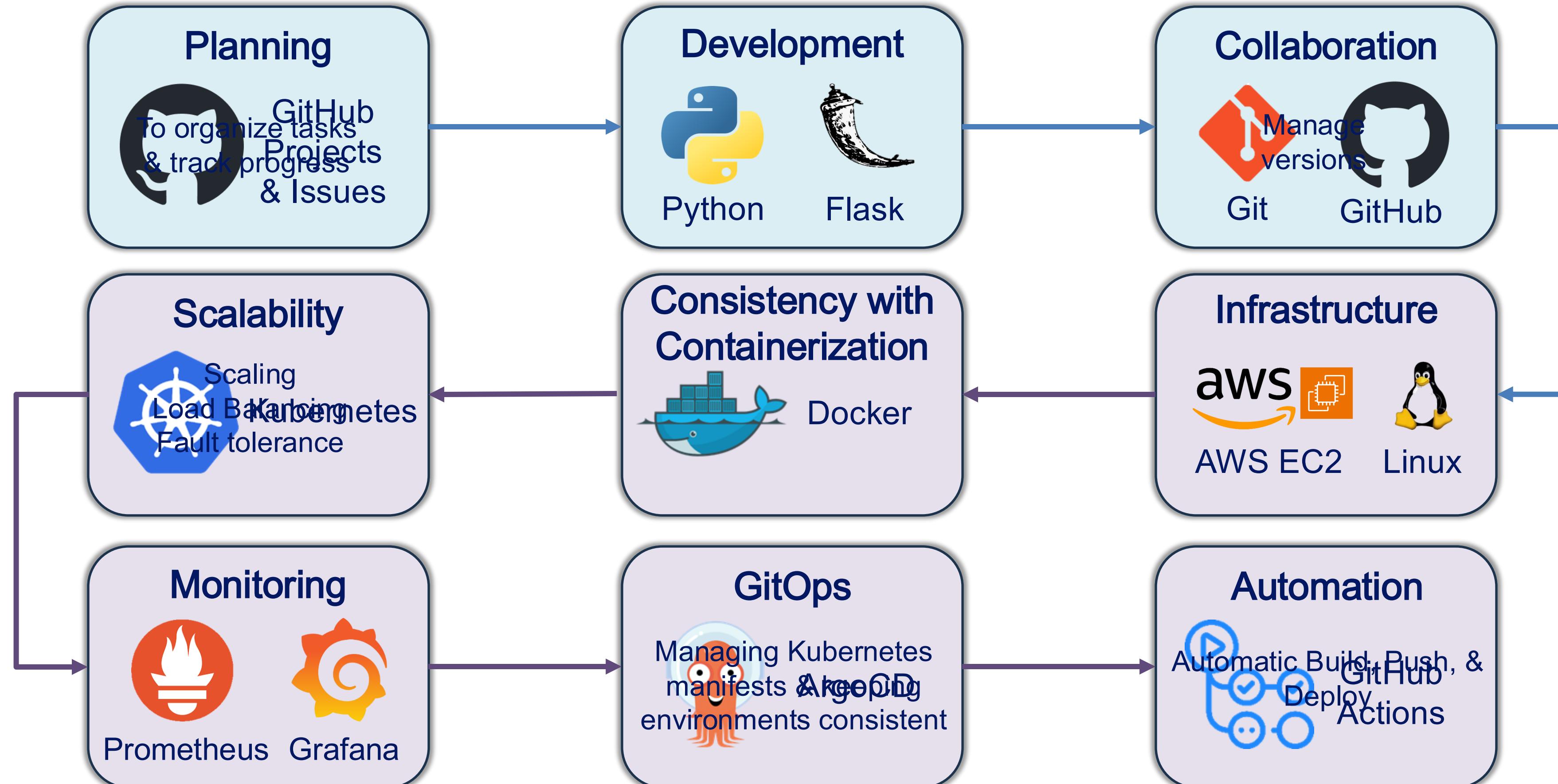
- ✓ Book App – Easy browsing and ordering
- ✓ Agile Delivery – Fast releases without breakage
- ✓ Scalable System – Reliable, monitored, and growth-ready

Challenge



- ✓ Automation
- ✓ Monitoring
- ✓ Scalability

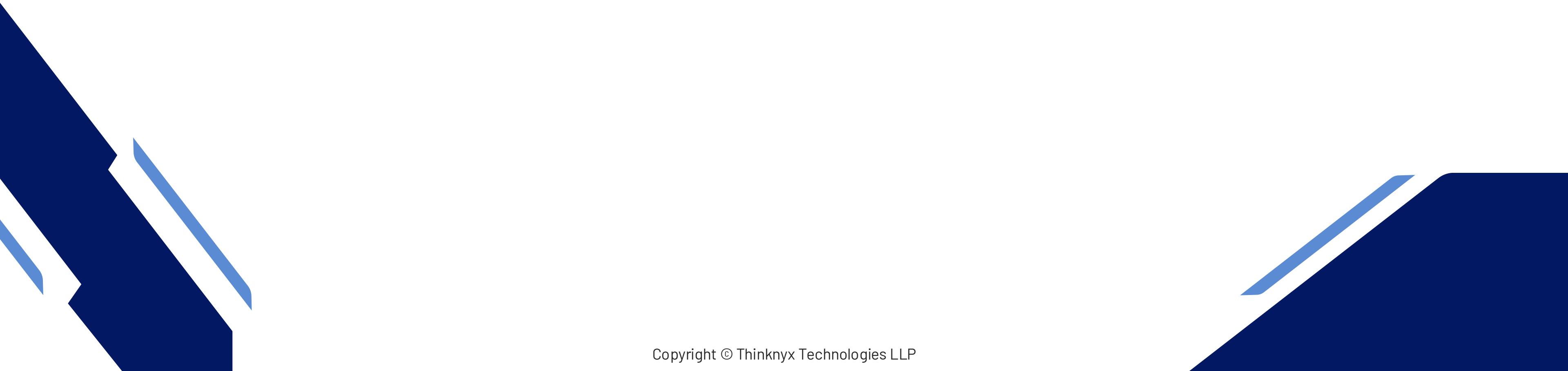
Course Goal



Section 1

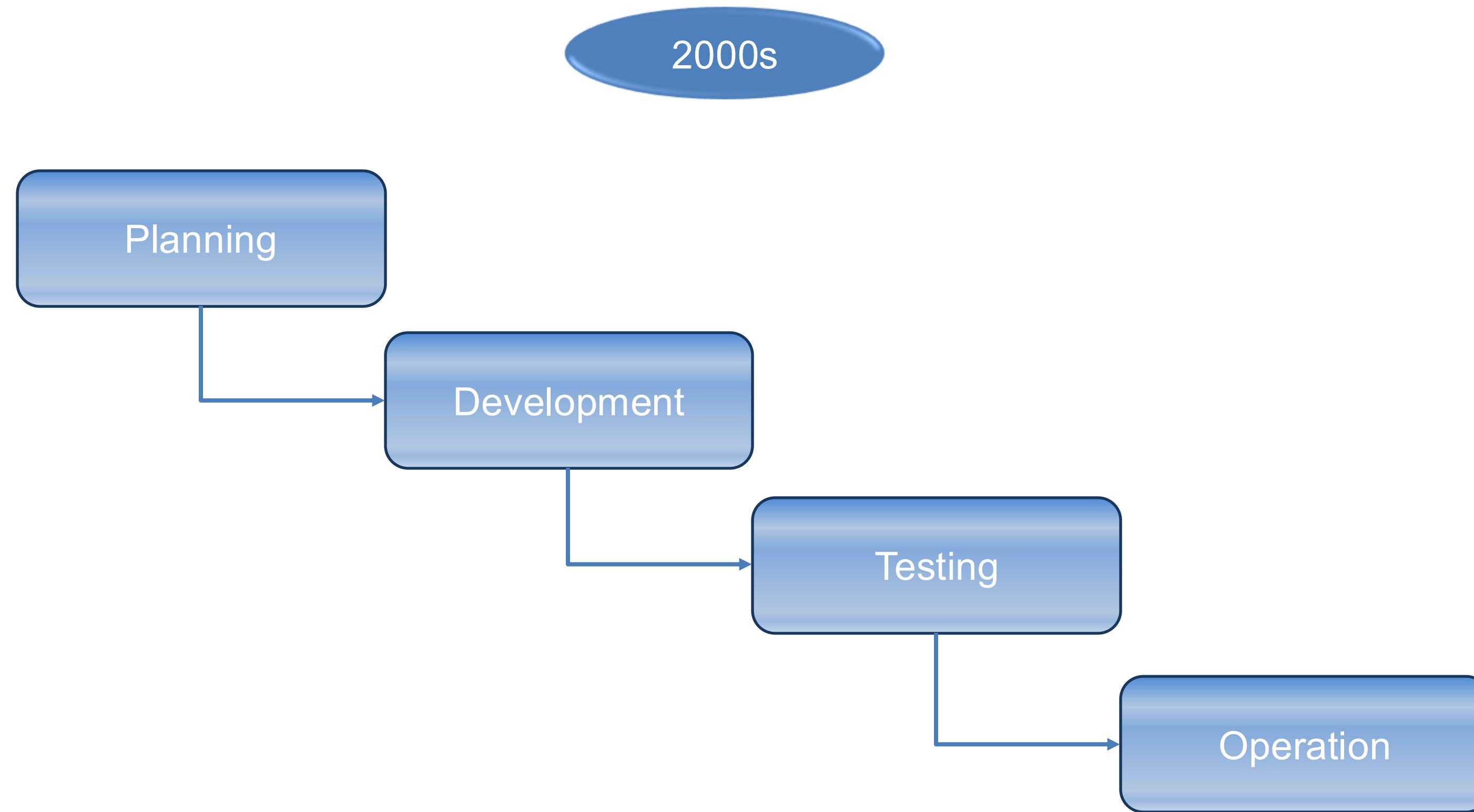
DevOps Course

Presented By: Thinknyx



What is DevOps? Evolution and Benefits

The Waterfall Model



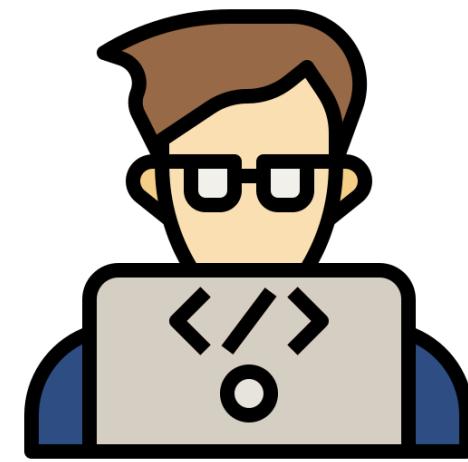
Development and Operations Teams

Development Team

Operations Team

Development and Operations Teams

Development Team

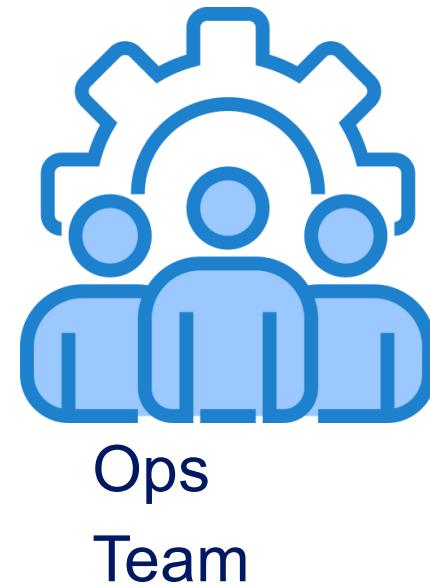


Developer

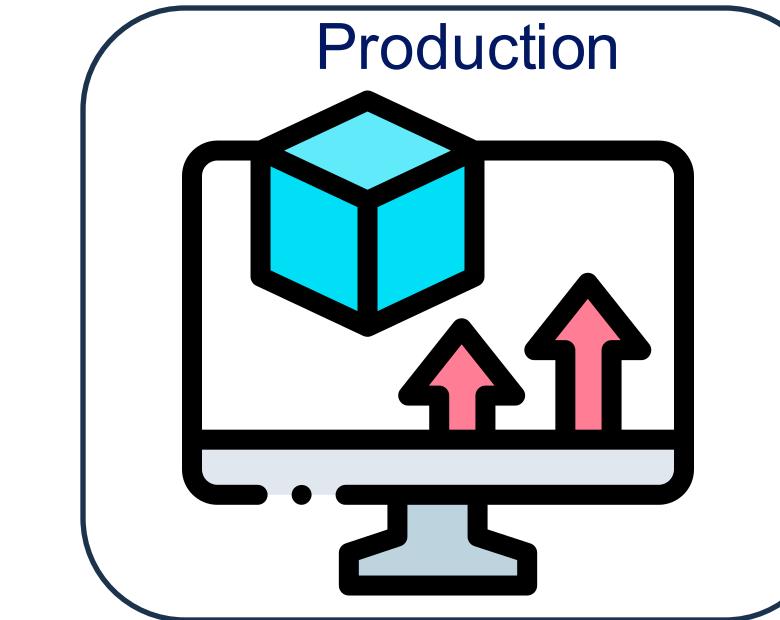


- ✓ Writing code
- ✓ Fixing bugs
- ✓ Designing architectures
- ✓ Running unit
- ✓ Integration tests
- ✓ Preparing builds for release

Development and Operations Teams



- ✓ Stable
- ✓ Secure
- ✓ Available



- ✓ Configuring servers
- ✓ Networks
- ✓ Monitoring uptime
- ✓ Performance
- ✓ Handling incidents

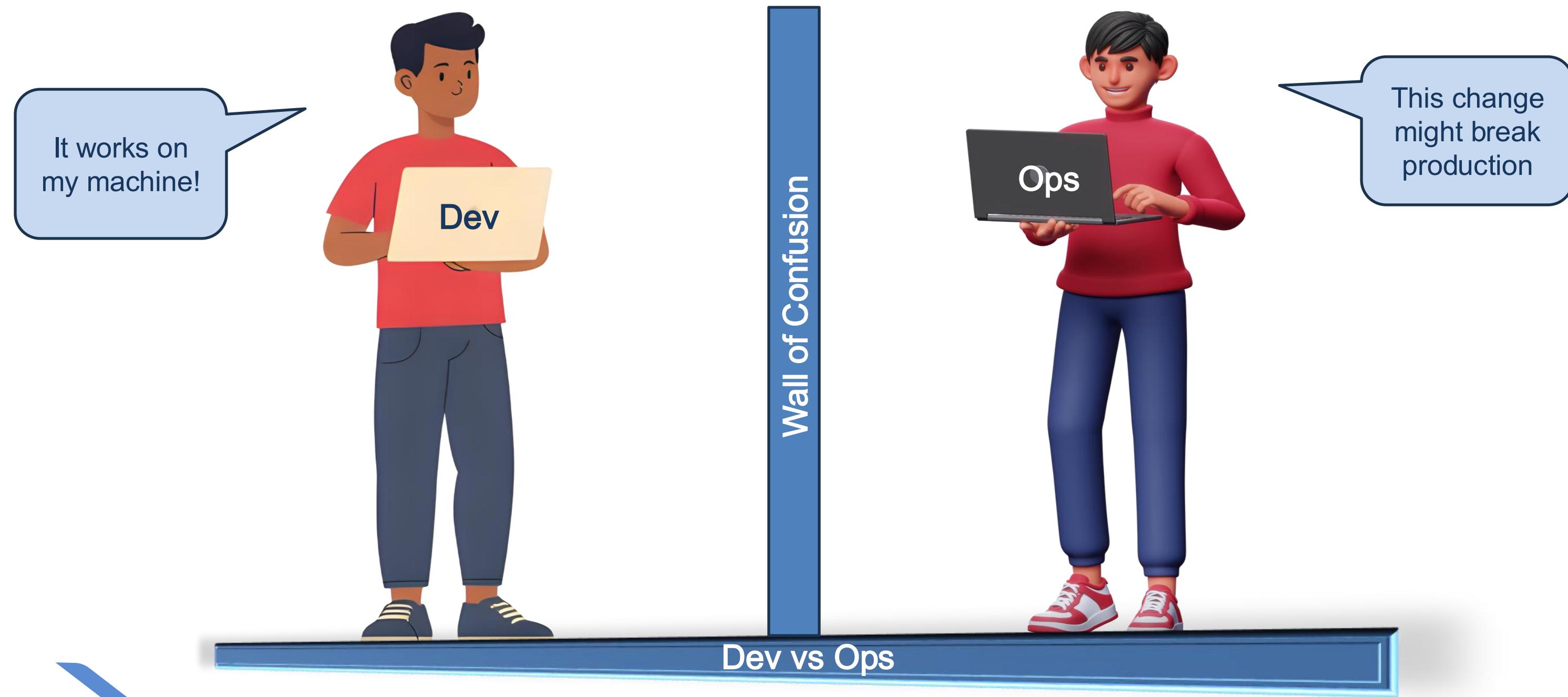
Focus wasn't speed - it was **stability** and **reliability**

Smooth-running systems

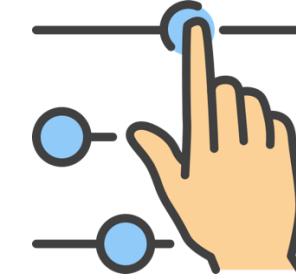
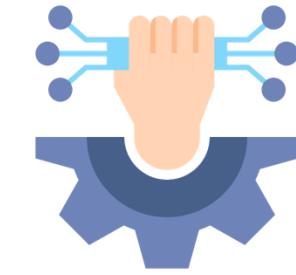
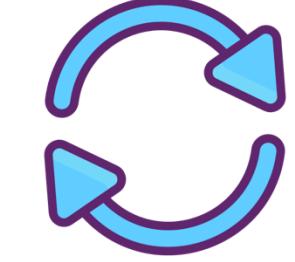
Minimal downtime

The Dev vs Ops Wall

- Developers were pushing for **speed**
- Operations fought for **stability**



The Agile Model



Agile emerged
in 2001

Faster
deliveries &
shorter time-
to-market

Iterations,
collaboration
, feedback
loops

Revolutionized
development

Rapid dev,
lagged

Slow, error-
prone
deployments

Manual
infrastructur
e

Dev
boosted,
Ops gap
persisted

The Birth of DevOps

2007



Patrick

- Challenging
Development & Operations

2008



Andrew

- Concept of Agile Infrastructure
Sharing

2009: Organized first DevOpsDays in Ghent, Belgium

What is DevOps



Goal

Bridge the gap between Dev & Ops



Principles

Agile, automation, collaboration



Responsibility

Focus on speed & stability



Success

Balance delivery & reliability



Early Ops

Deployment from the start



Beyond Workflow

DevOps is a culture



Core Elements

Philosophies, practices, tools



Outcome

High-velocity delivery



Impact

Break barriers between Dev & Ops

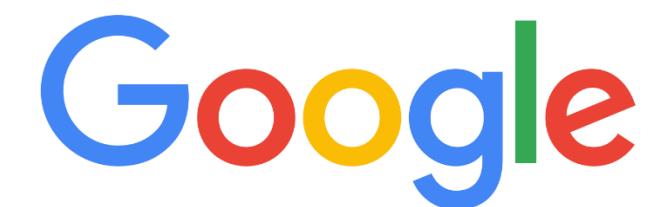
Companies and Their DevOps Outcomes



- Netflix – Built their own platform Spinnaker
- Enables thousands of changes daily
- Achieved with zero downtime



- Amazon – Over 50 million deployments per year
- 136,000 deployments per day
- Almost 2 deployments per second
- Demonstrates extreme speed & reliability



- Google & Others – Constant deployments
- Across hundreds of microservices

Measuring Success with DORA Metrics

Deployment Frequency

Lead Time for Changes

Change Failure Rate

Mean Time to Recovery

Before and After DevOps

➤ Before DevOps

- Teams worked in silos (Dev vs Ops)
- Inconsistent environments
- Manual, error-prone deployments
- Slow scaling of systems
- Resource wastage
- Speed vs stability conflict

➤ After DevOps

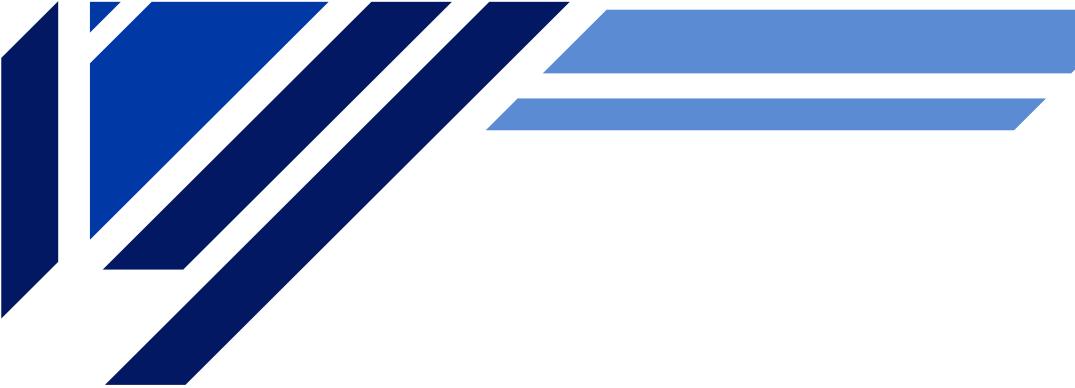
- Collaboration from the start
- Containers & CI/CD ensure consistency
- Automation replaces manual tasks
- Kubernetes enables on-demand scaling
- Cloud & IaC optimize resources
- Shared responsibility for speed & reliability

DevOps Difference

From slow, fragile, siloed → To fast, automated, collaborative

Challenges in Traditional Organizations

- Traditional organizations often rely on old habits
- Teams stuck in silos resist change
- Misconception: new tools alone can “do DevOps”
- DevOps is more about people and culture than tools
- Success requires understanding the whole value stream (idea to end-user)
- Collaboration across the entire product lifecycle is key
- Breaking silos enables real teamwork and innovation



DevOps Lifecycle

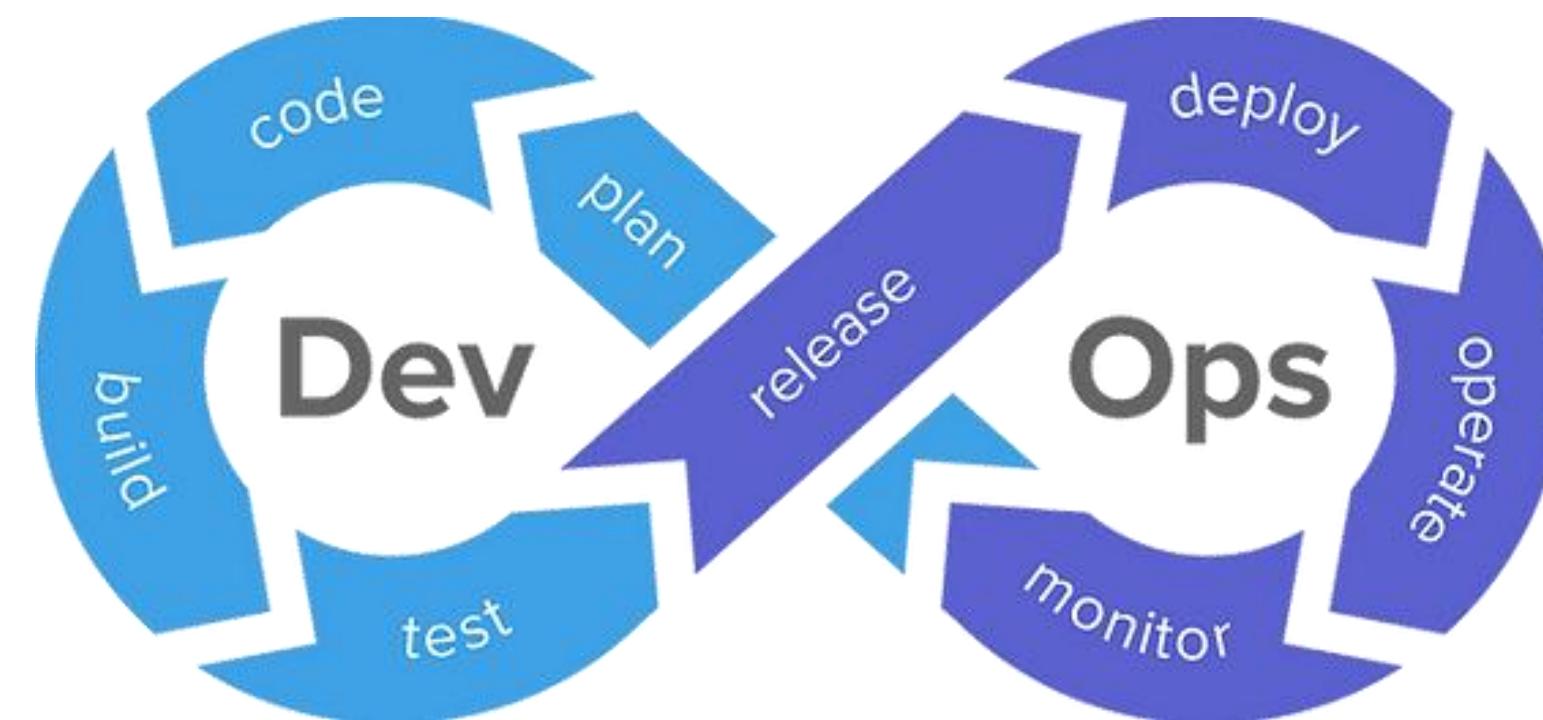


DevOps Lifecycle

- A set of continuous processes

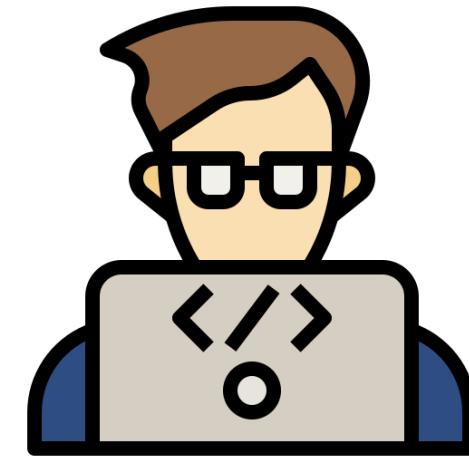


- Ensure speed and quality



DevOps Lifecycle

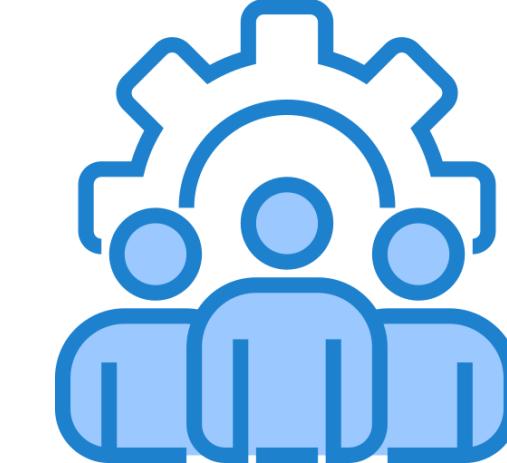
➤ Plan



Developers



Testers



Operations



Business stakeholders

- ✓ New features
- ✓ Bug fixes
- ✓ Improvements

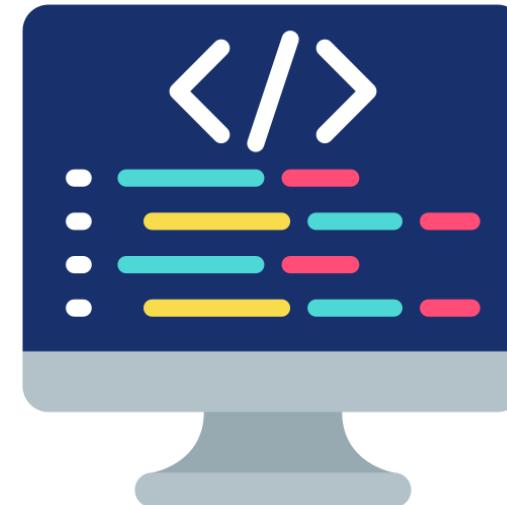
They rely on user feedback, business goals, and technical requirements

Prioritized backlog

MVP
(Minimum Viable Product)

DevOps Lifecycle

➤ Code



Develop

Write codebase after planning

Manage

Handle source code

Secure

Ensure maintainability & safety

Collaborate

Use tools & Git

Parallel

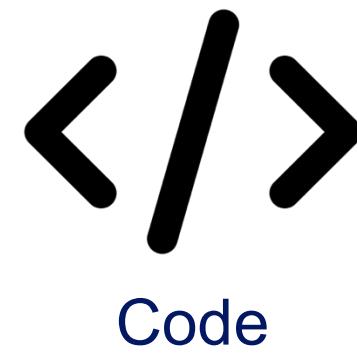
Avoid conflicts

Deliver

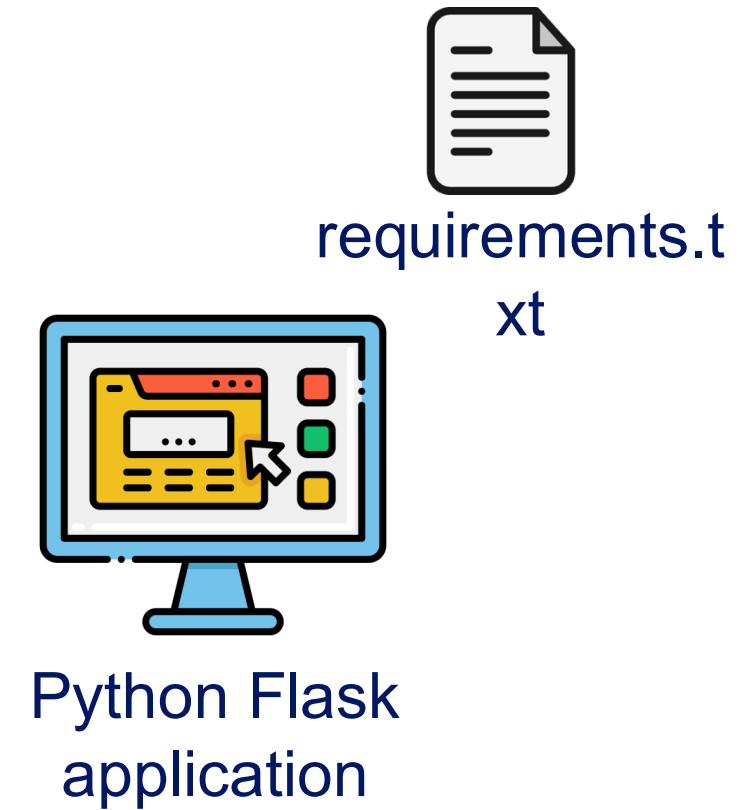
Turn ideas into software

DevOps Lifecycle

➤ Build



Code



DevOps Lifecycle

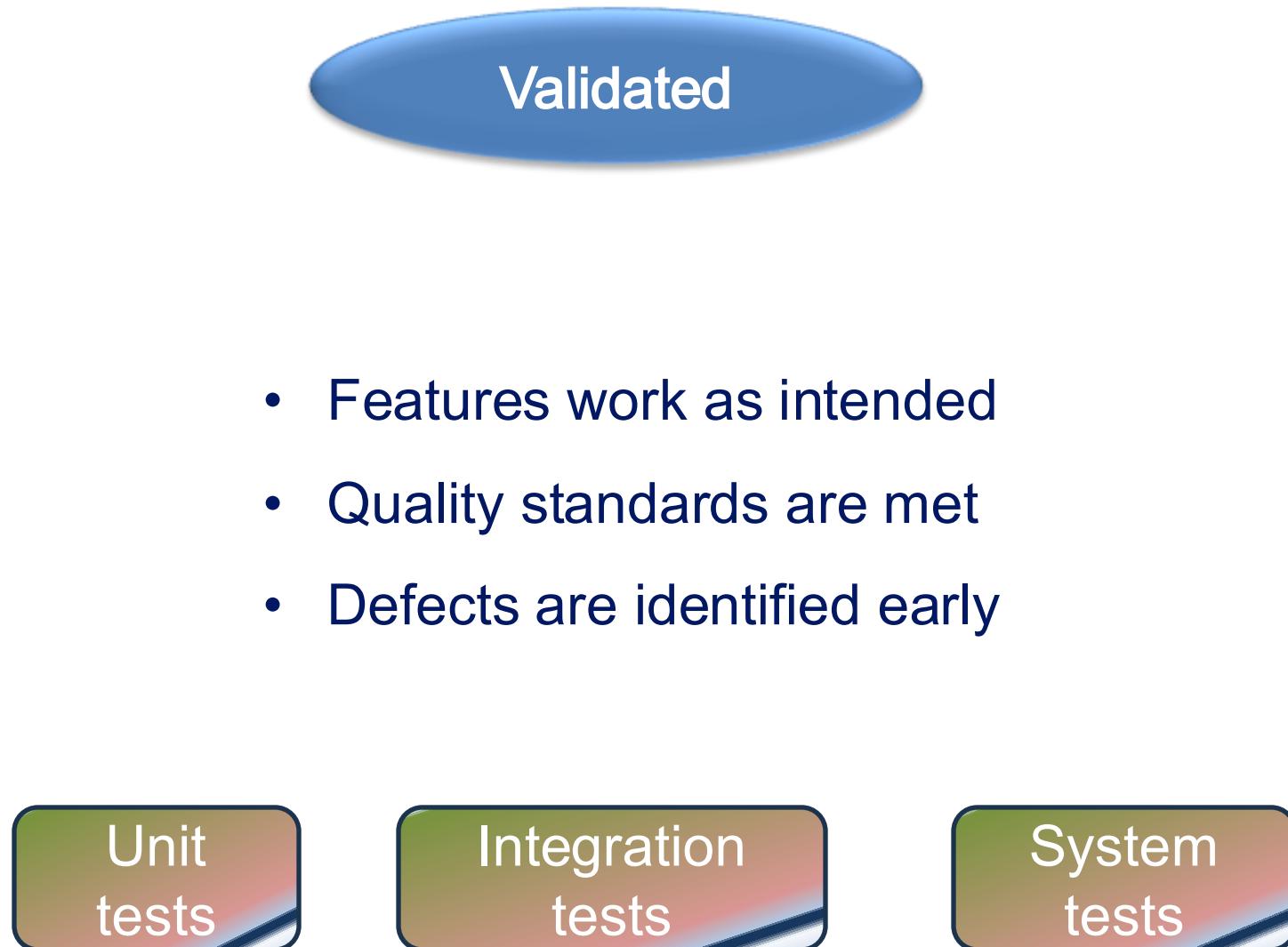
➤ Build



- Ensures consistency across environments (developer laptop, testing, production)

DevOps Lifecycle

➤ Test



- Features work as intended
- Quality standards are met
- Defects are identified early

- Testing performed in test/staging environment
- Environment closely mimics production setup
- Helps identify and fix issues before release
- Builds confidence for final production deployment
- Ensures application is stable and reliable

DevOps Lifecycle

➤ Release

- Application prepared for release
- Software and dependencies bundled into a release candidate
- Candidate is **stable, secure, and requirement-compliant**
- Infrastructure components (**servers, databases, load balancers**) provisioned
- Final polishing before availability to end users

DevOps Lifecycle

➤ Deploy

Deploy

Application moved
to production
environment

Canary Release

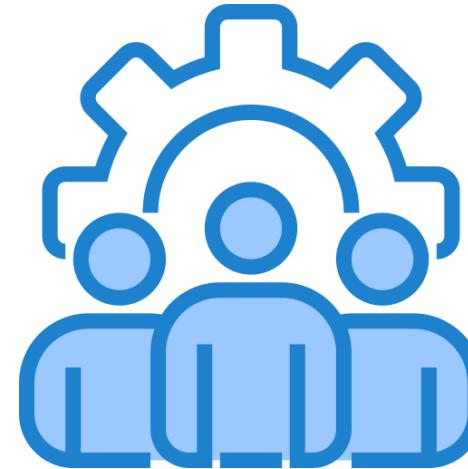
New features rolled
out to a subset of
users first for stability
testing

Significance

Marks the point
where planning and
coding reach real
users

DevOps Lifecycle

➤ Operate



Operation
s

Ensure the application runs smoothly

- ✓ High availability
- ✓ Scalability
- ✓ Performance



□ SLAs (Service Level Agreements)

- Expected level of service
 - Uptime: 99.9% availability per month
 - Response time: Average API call should respond in <200 ms
 - Support: Critical incidents resolved within 2 hours
- Ensure product meets agreed service levels (SLAs)

DevOps Lifecycle

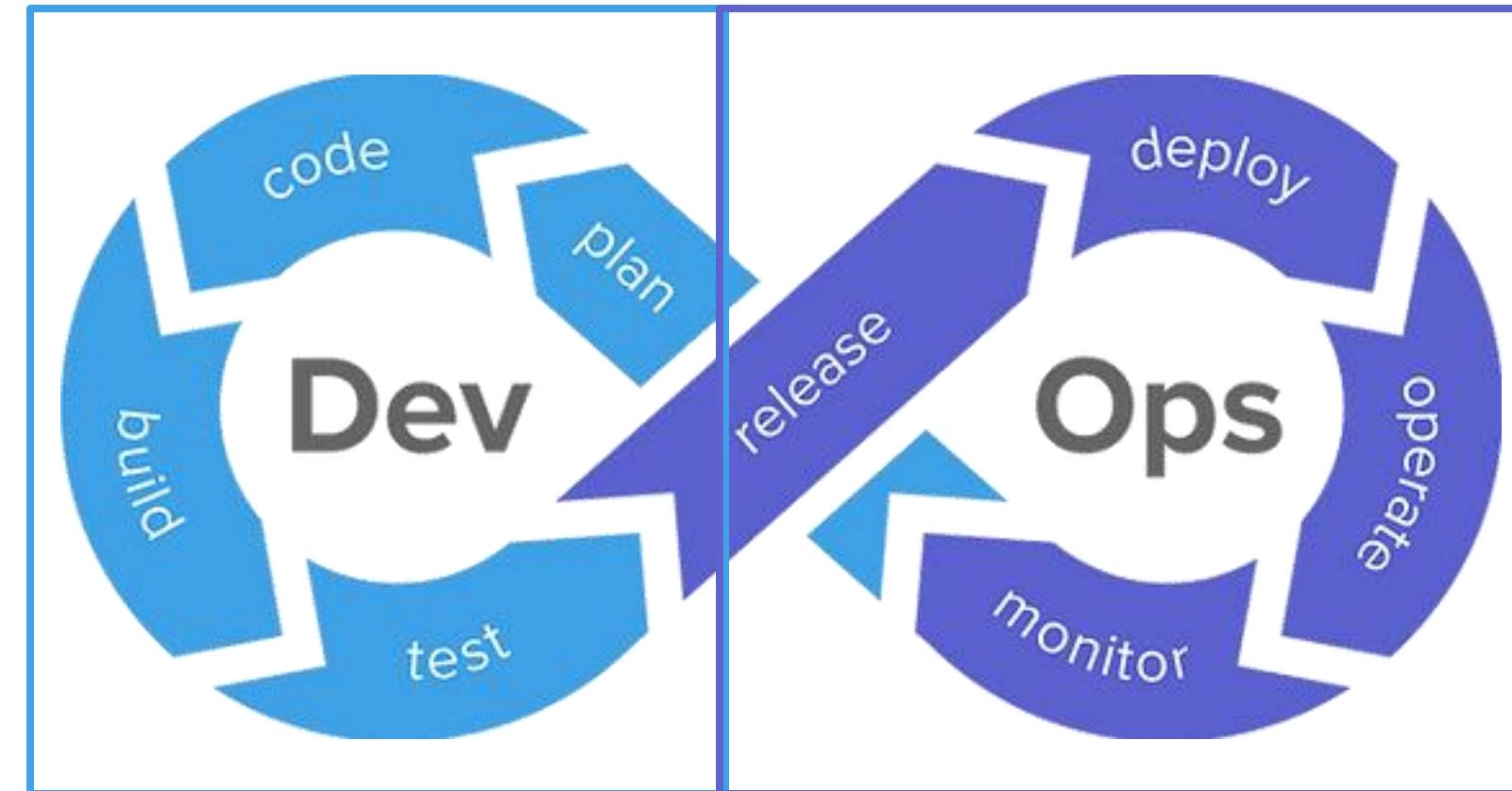
➤ Monitor

- Applications are continuously monitored
- Performance metrics, error logs, and customer feedback are tracked
- Monitoring ensures SLAs are honoured
- Alerts trigger immediate action if availability/performance dips
- Insights from users and system behaviour feed into planning
- Supports continuous improvement cycle

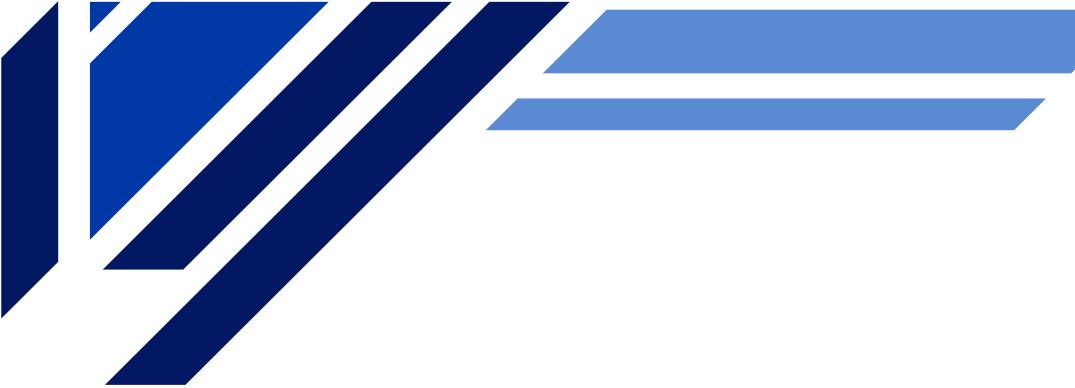
DevOps Lifecycle

- Continuous infinity loop (∞)

Create, improve, and validate the software



Ensure smooth delivery, availability, performance, and reliability in live environments



CI/CD and Continuous Operations



CI/CD and Continuous Operations

CI/CD: Continuous Integration & Continuous Delivery or Deployment

Principle

Frequent, small updates instead of big releases

Benefits

Reduced risk, easier issue detection, faster innovation

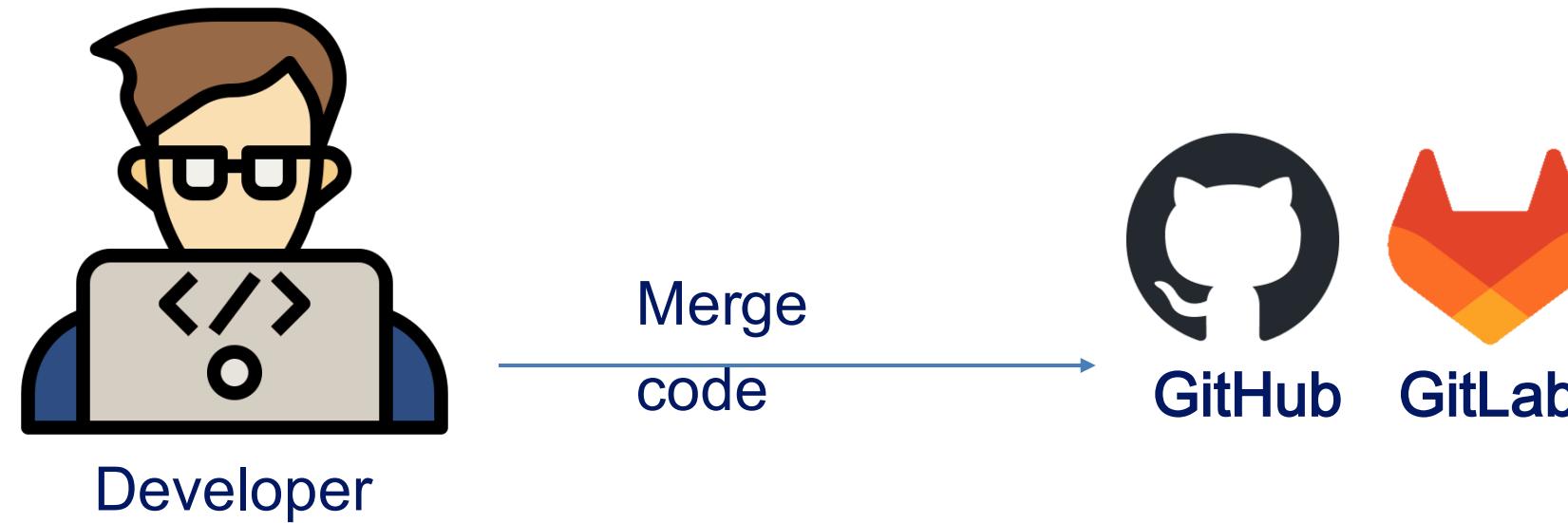
Microservices

Break large apps into smaller, independent services

More Deployments means Harder Manual Management

CI/CD and Continuous Operations

➤ Continuous Integration (CI)



- Each time code is merged, automated builds and tests find bugs early
 - Improve code quality
 - Reduce the time

- Avoids painful "merge days"
- Continuous Integration (CI) enables daily integration
- Developers push small, frequent updates
- Automated tests provide quick feedback
- Ensures changes are safe and stable

CI/CD and Continuous Operations

➤ Continuous Delivery (CD)



- ✓ Built
- ✓ Tested
- ✓ Prepared for release

- Every code change passes automated checks
- Produces deployment-ready artifacts (e.g., container image, build package)
- Continuous Delivery ensures code is always deployable
- Final deployment triggered by operations team or release manager

CI/CD and Continuous Operations

➤ Continuous Deployment (CD)

- Every change passing automated tests is deployed directly to production
- No manual intervention required
- Users see updates within minutes
- Requires robust and reliable automated testing
- No manual gate before production

CI/CD and Continuous Operations

➤ Continuous Deployment (CD)

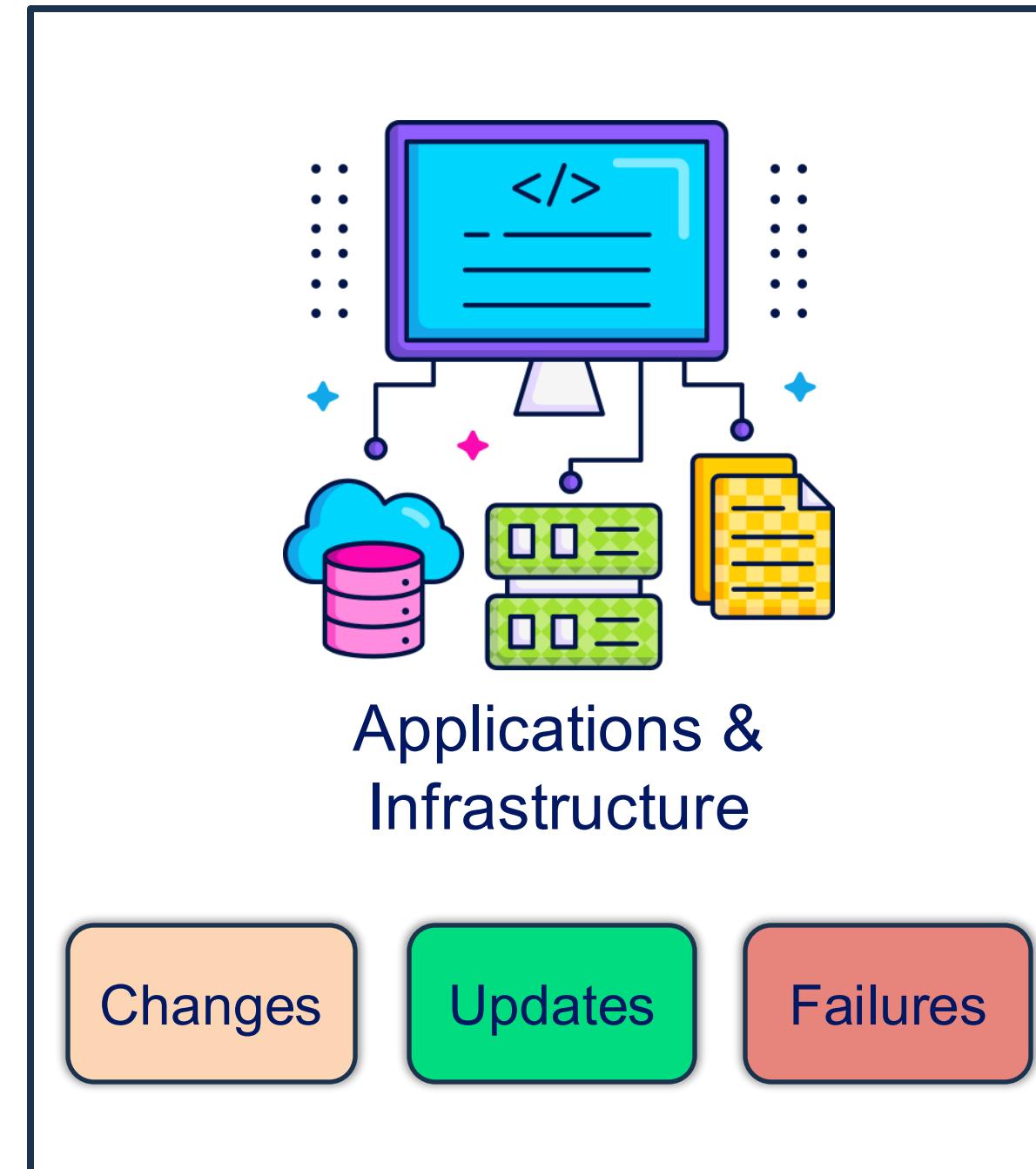


- Users see updates within minutes
- Requires robust and reliable automated testing
- No manual gate before production

Why CI/CD matters

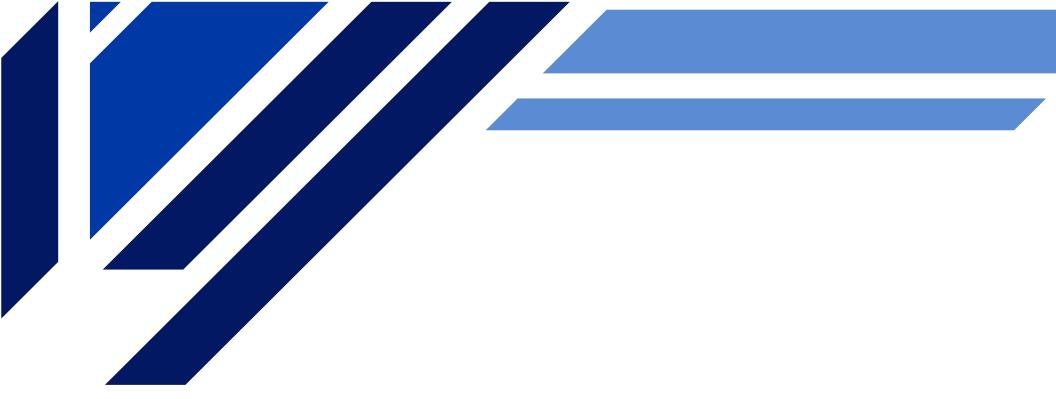
- Deliver features faster
- Reduce downtime
- Improve reliability
- Incorporate customer feedback more frequently

Continuous Operations



Operate Stage

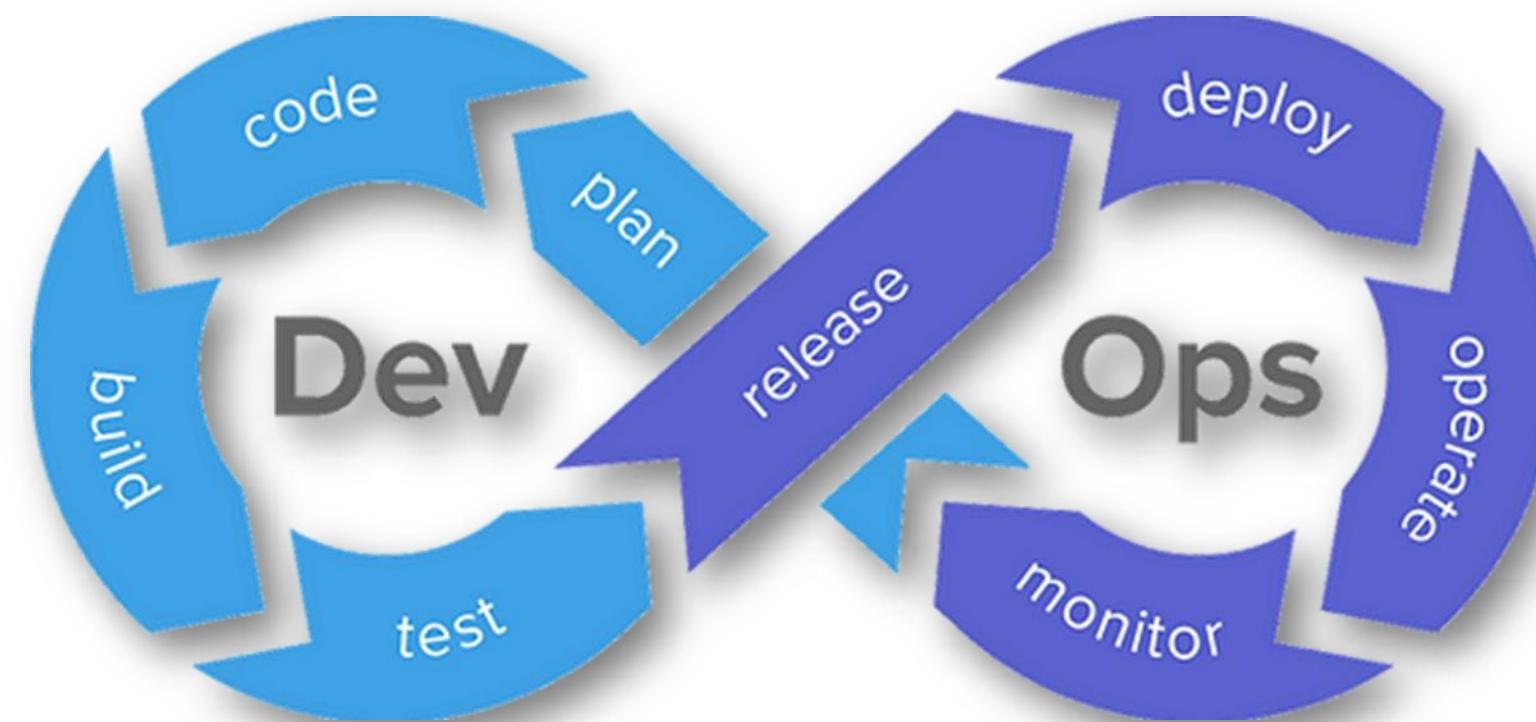
Monitor Stage



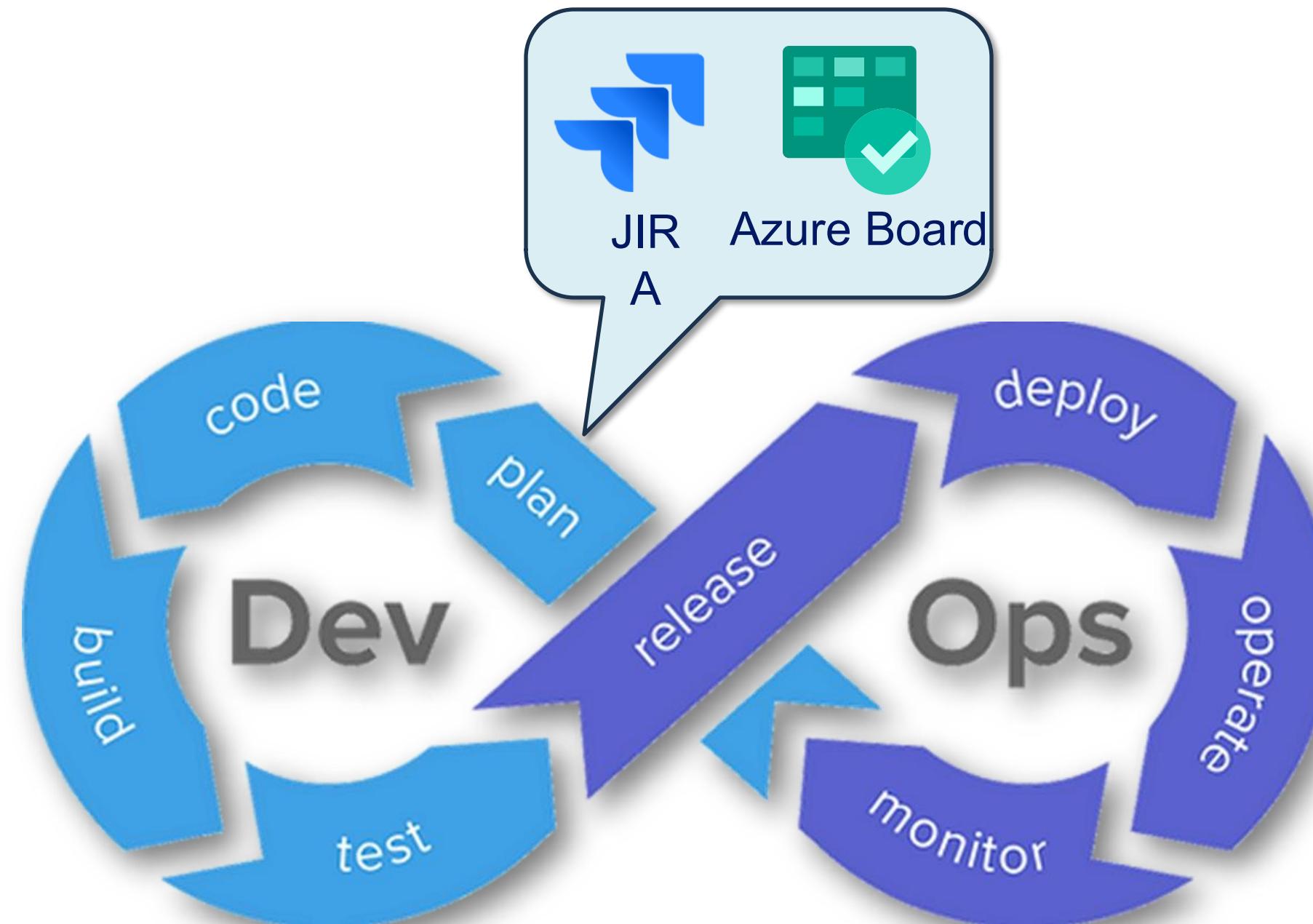
Setting up your Development Environment

Setting up your Development Environment

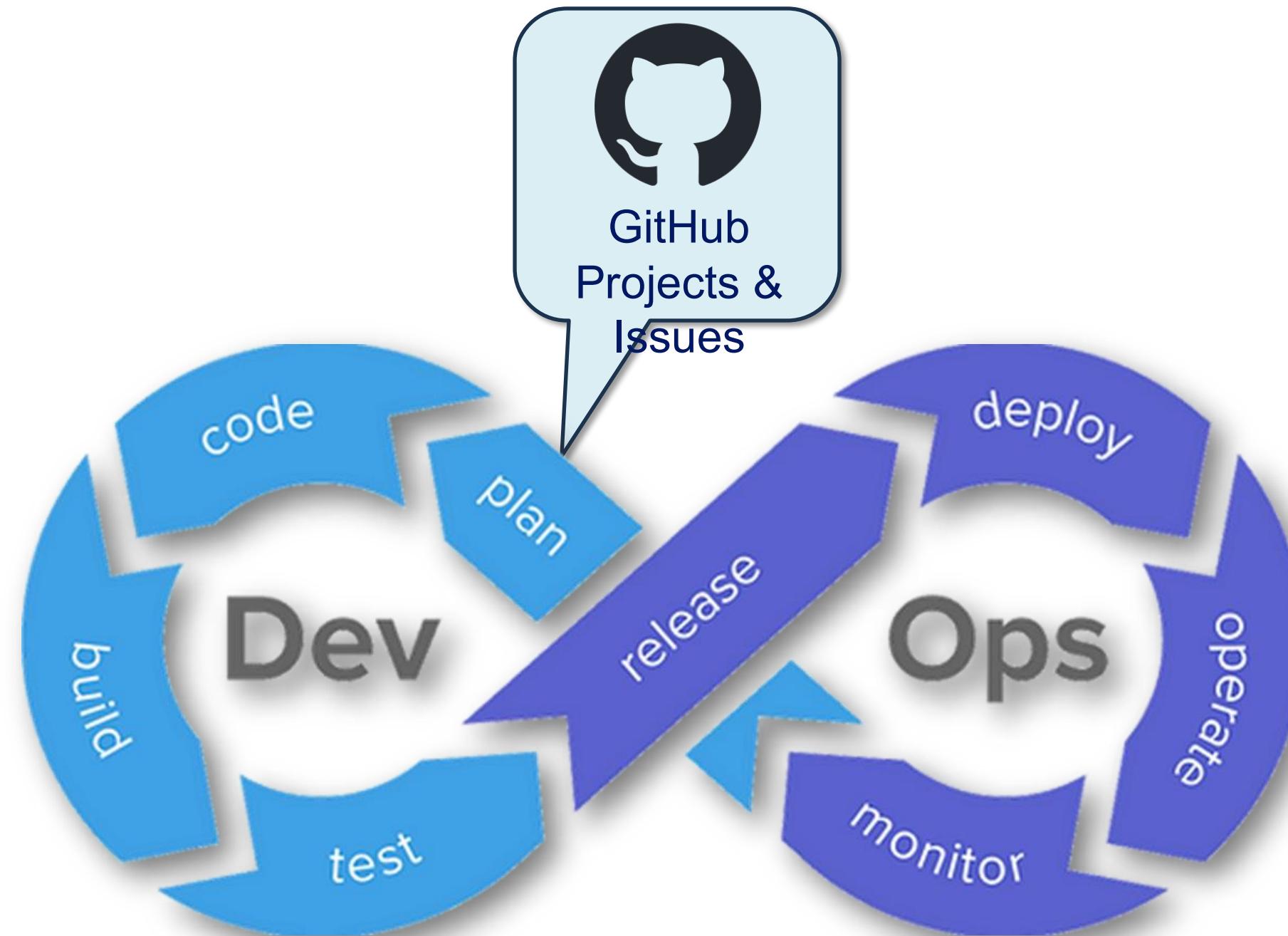
- DevOps tools



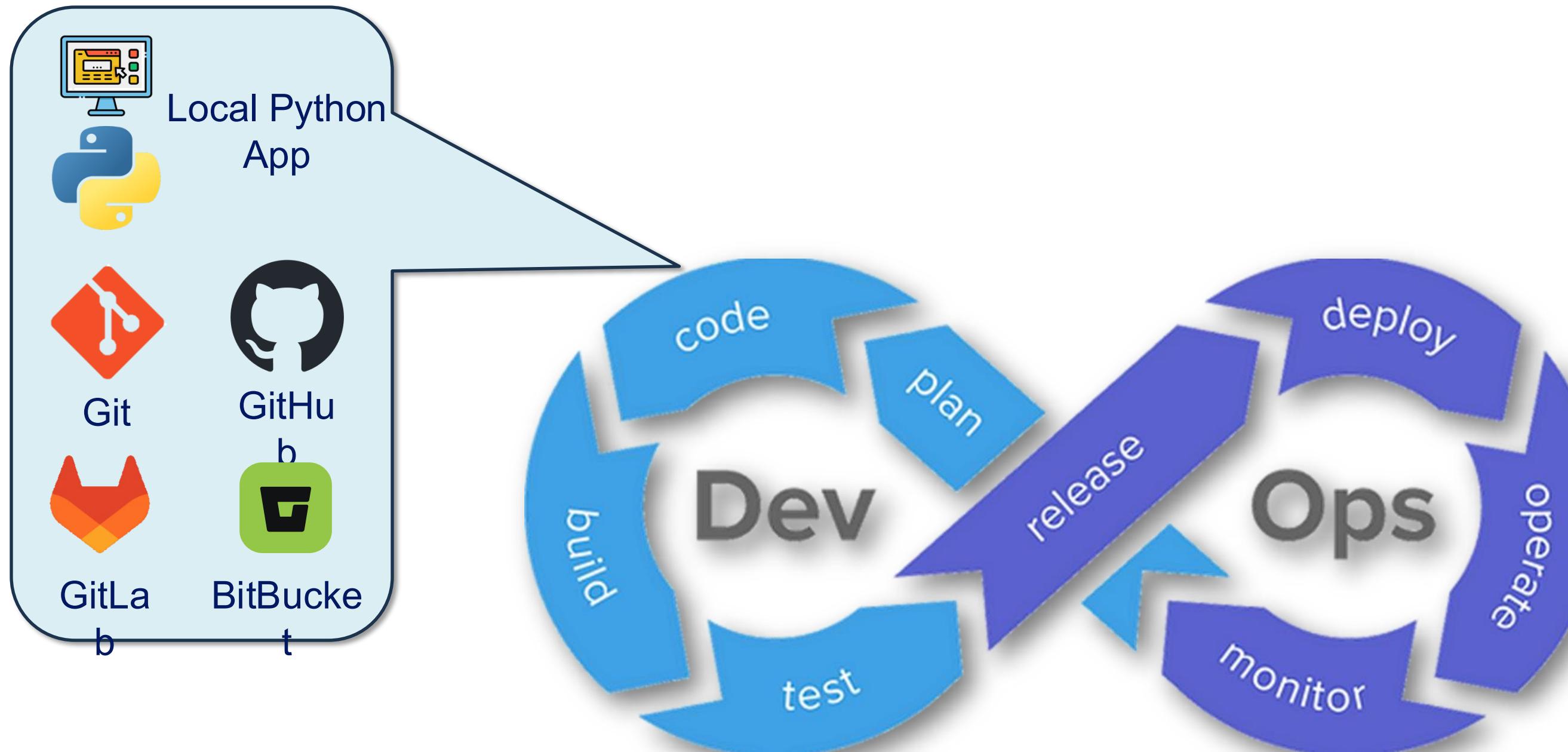
Setting up your Development Environment



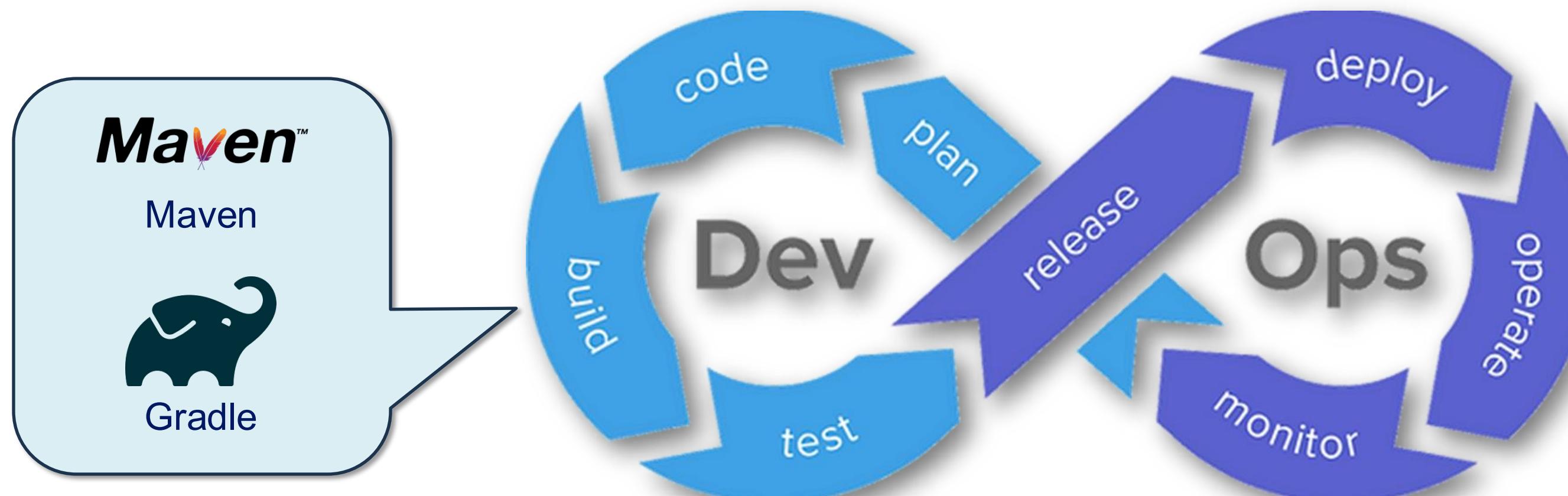
Setting up your Development Environment



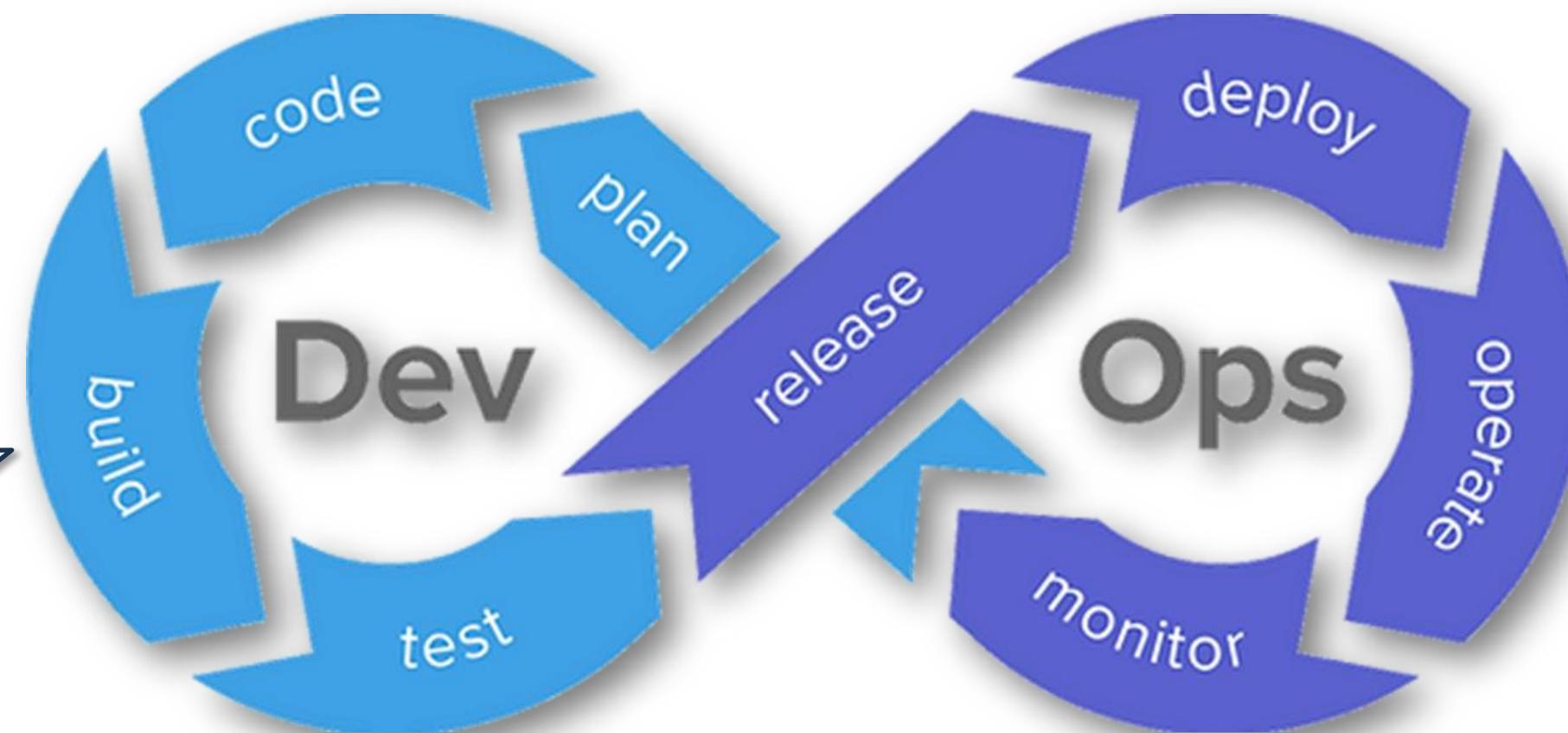
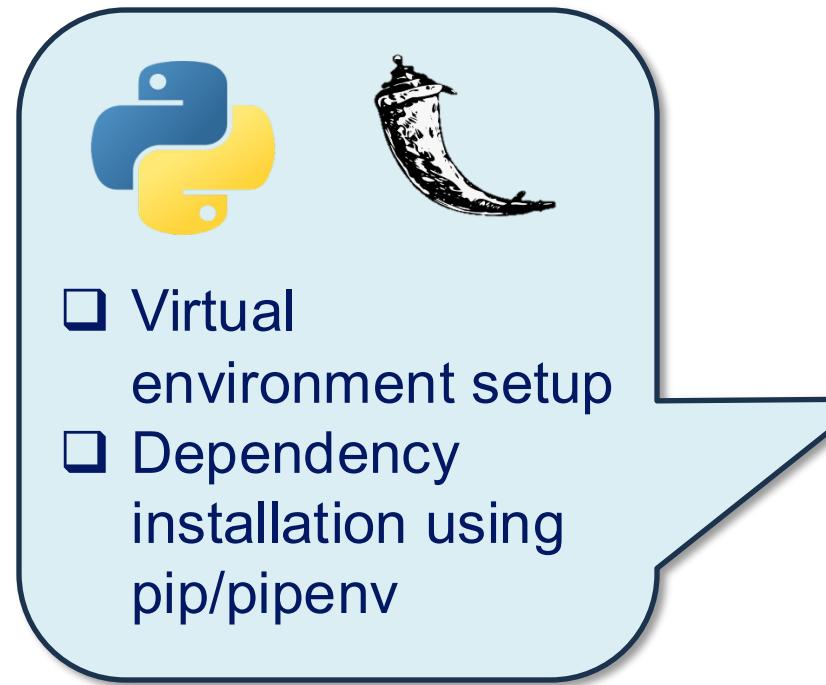
Setting up your Development Environment



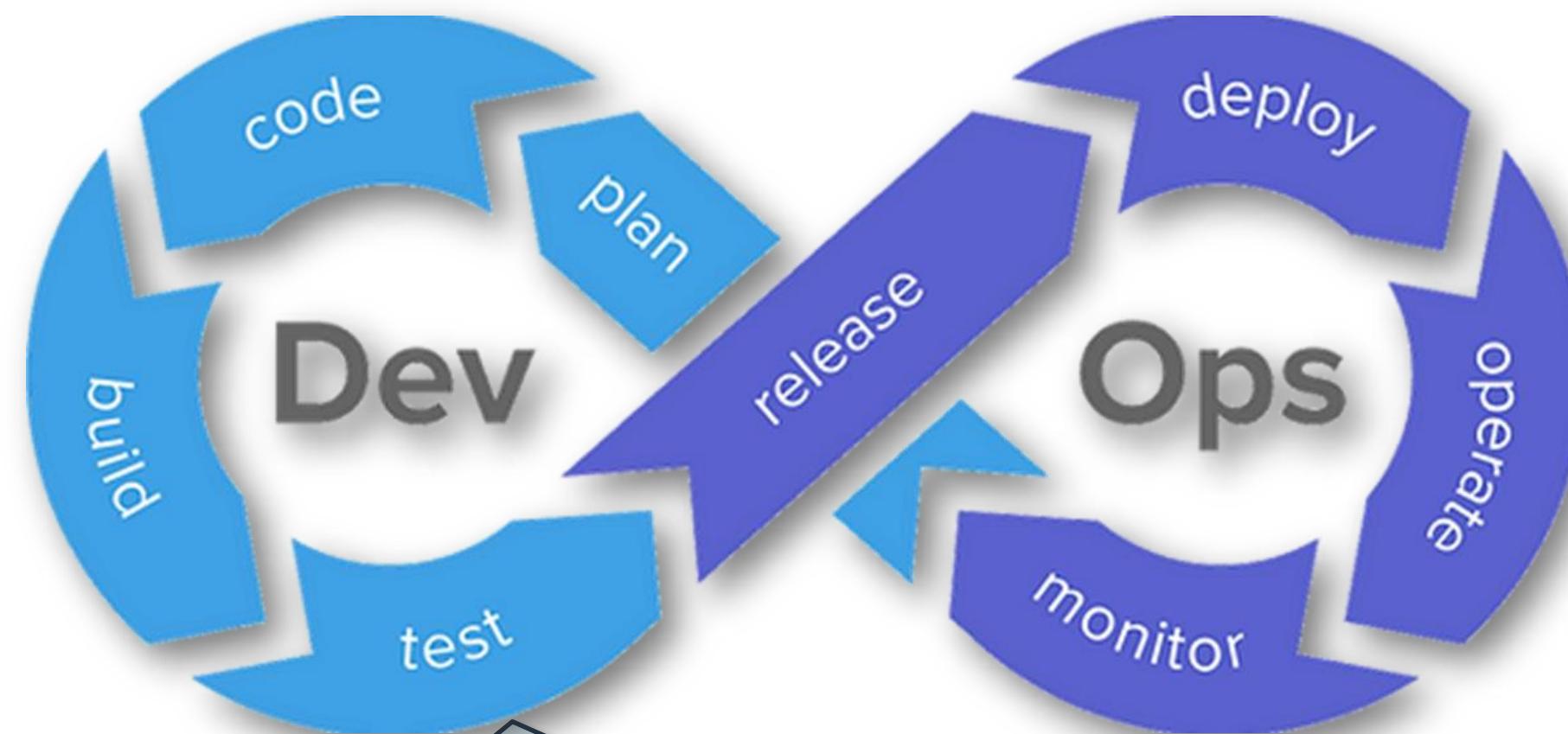
Setting up your Development Environment



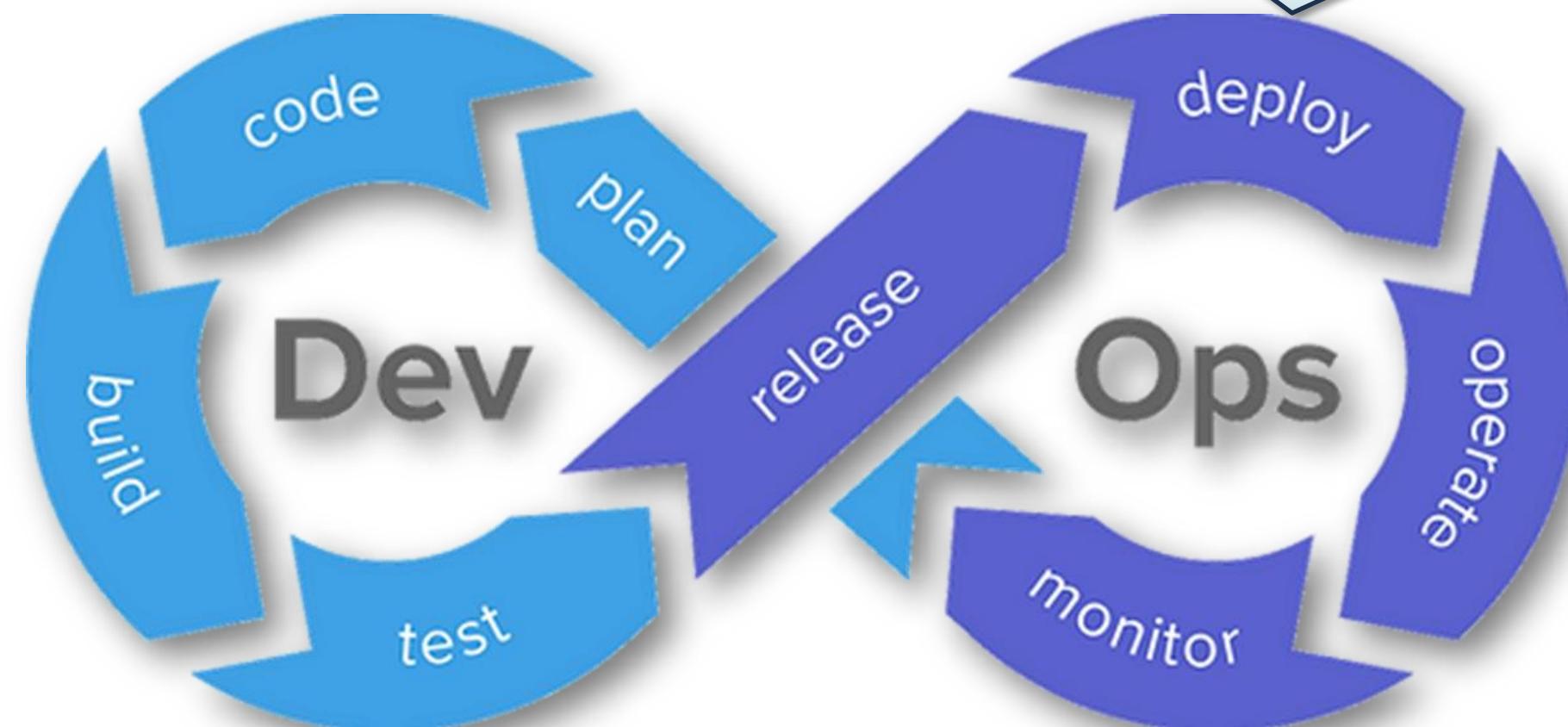
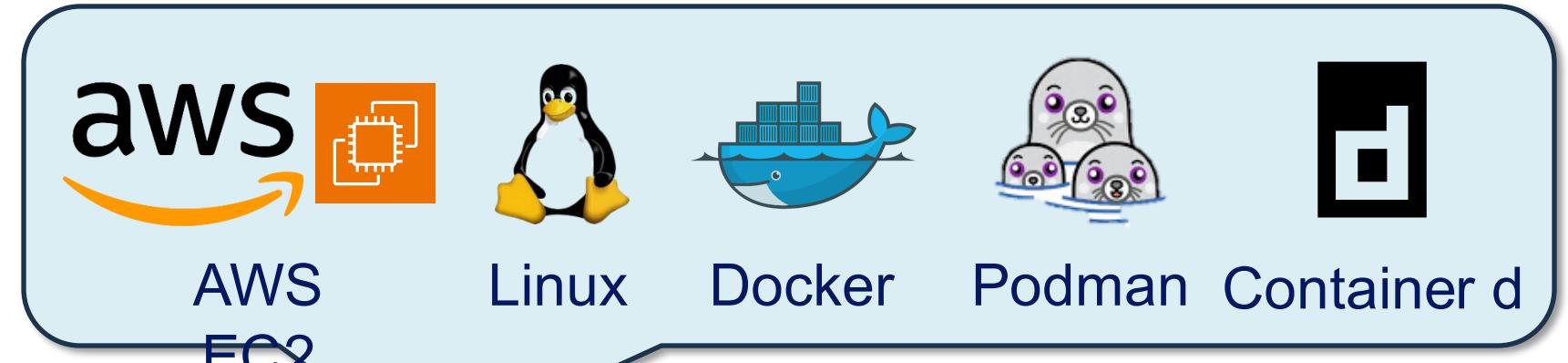
Setting up your Development Environment



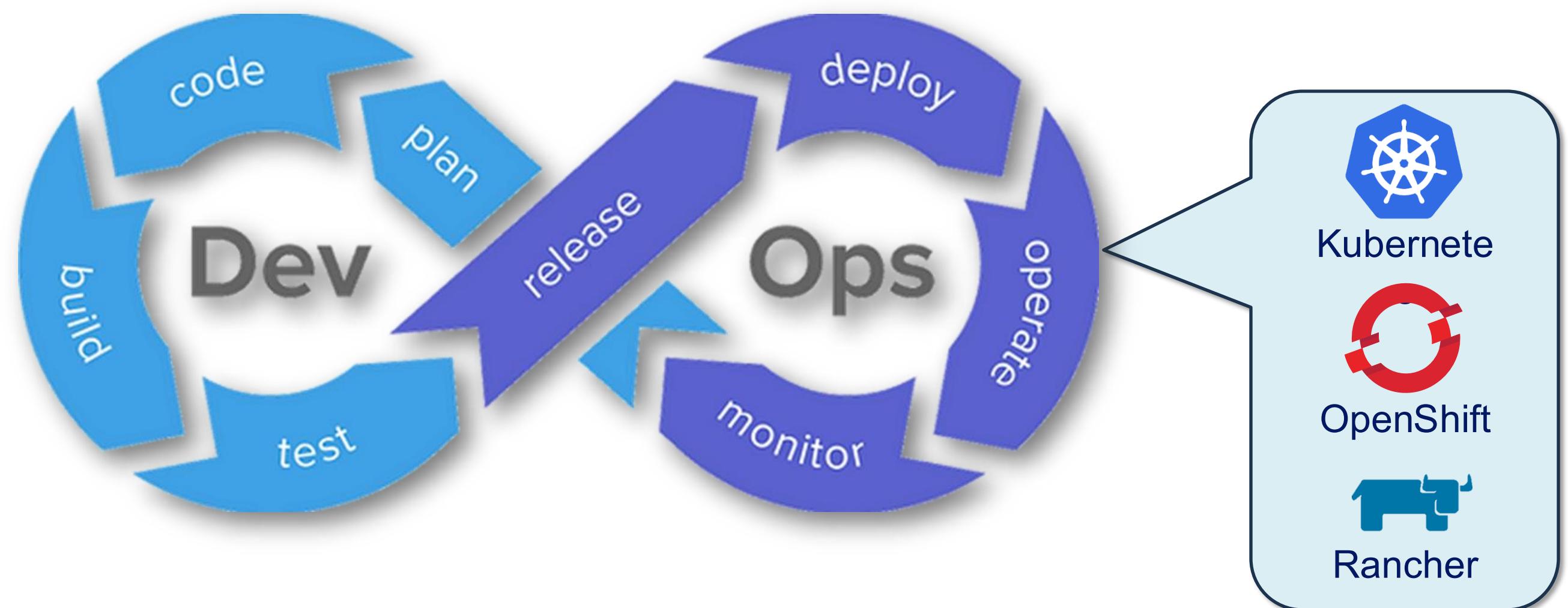
Setting up your Development Environment



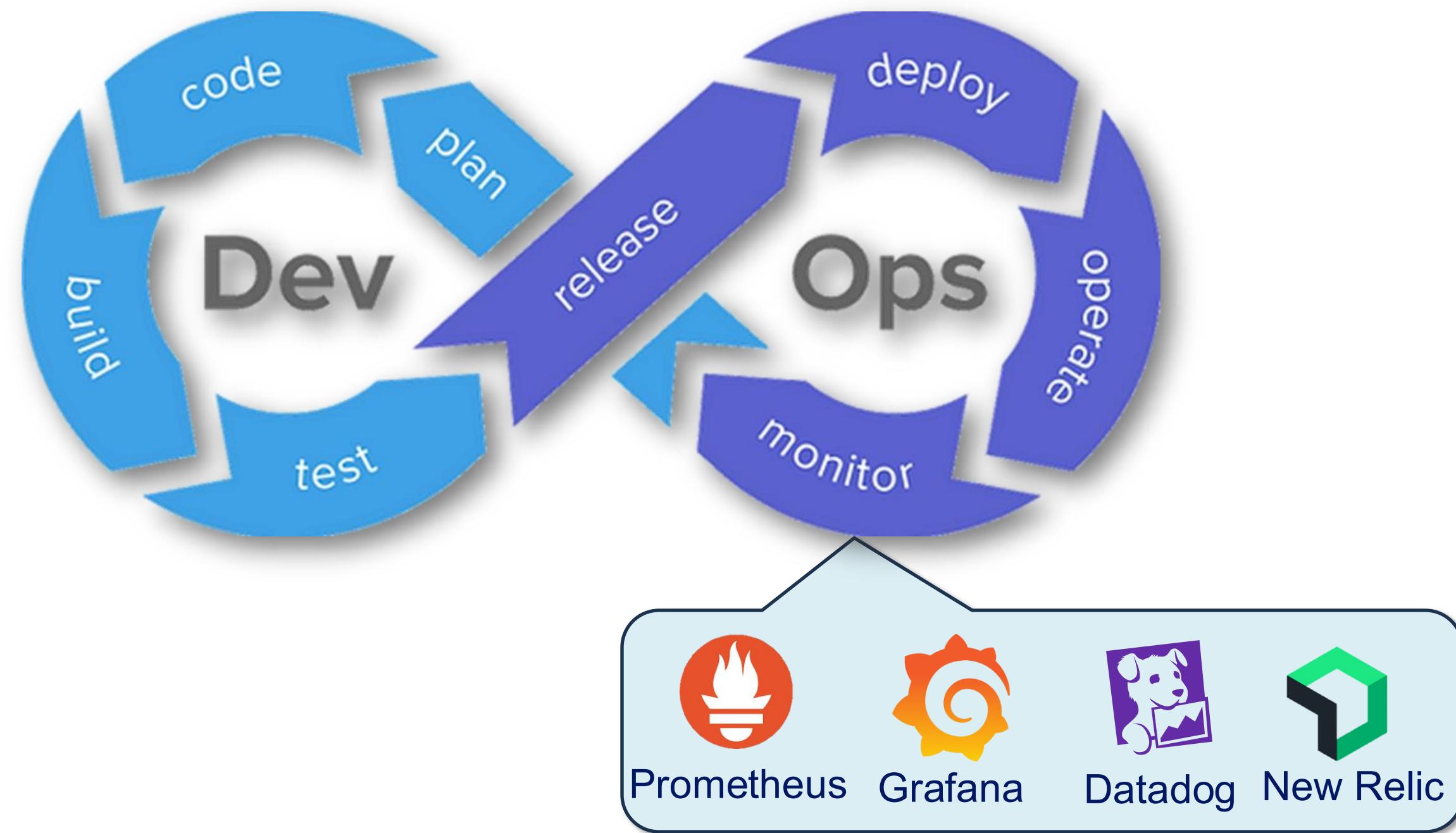
Setting up your Development Environment



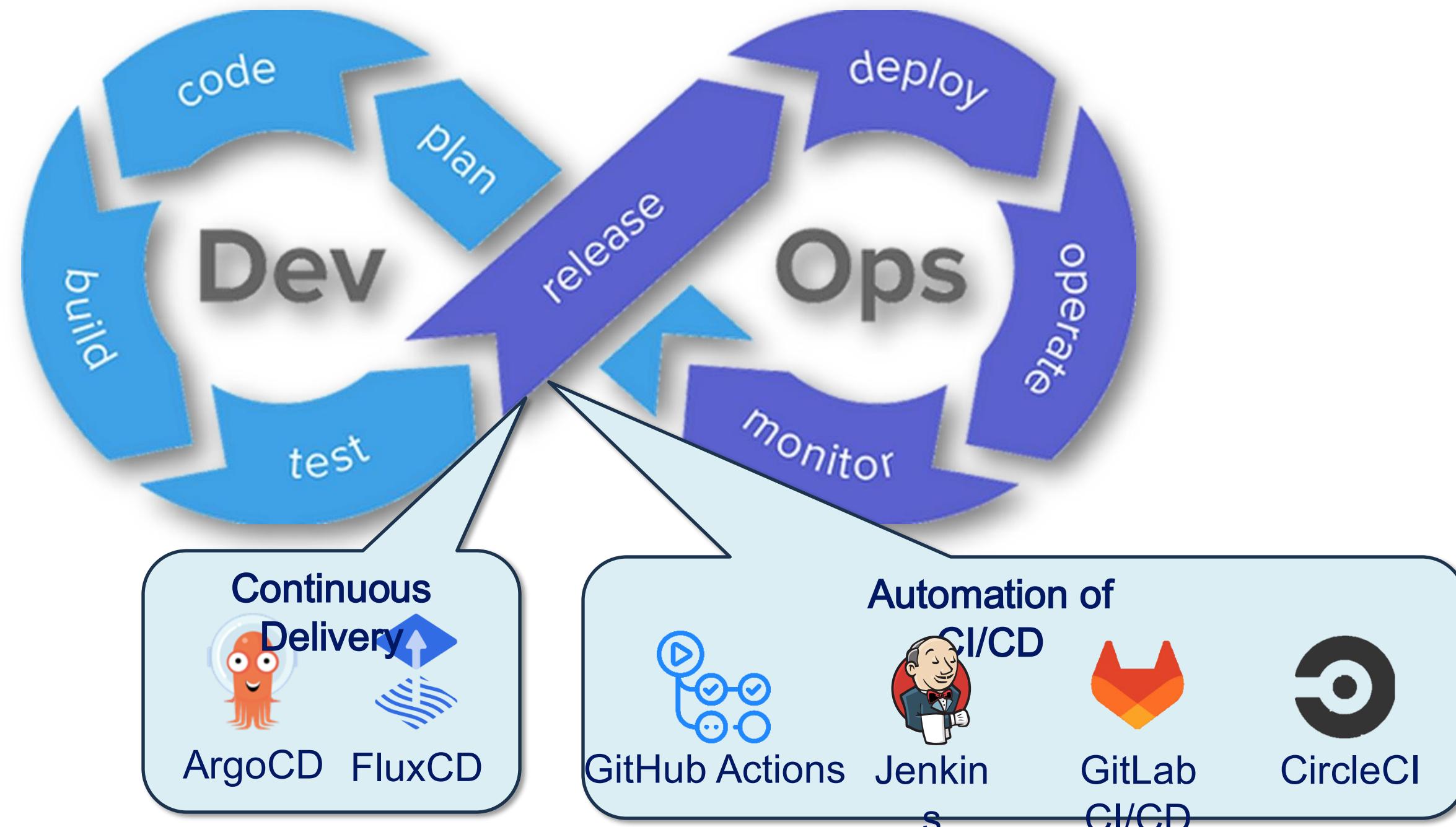
Setting up your Development Environment



Setting up your Development Environment



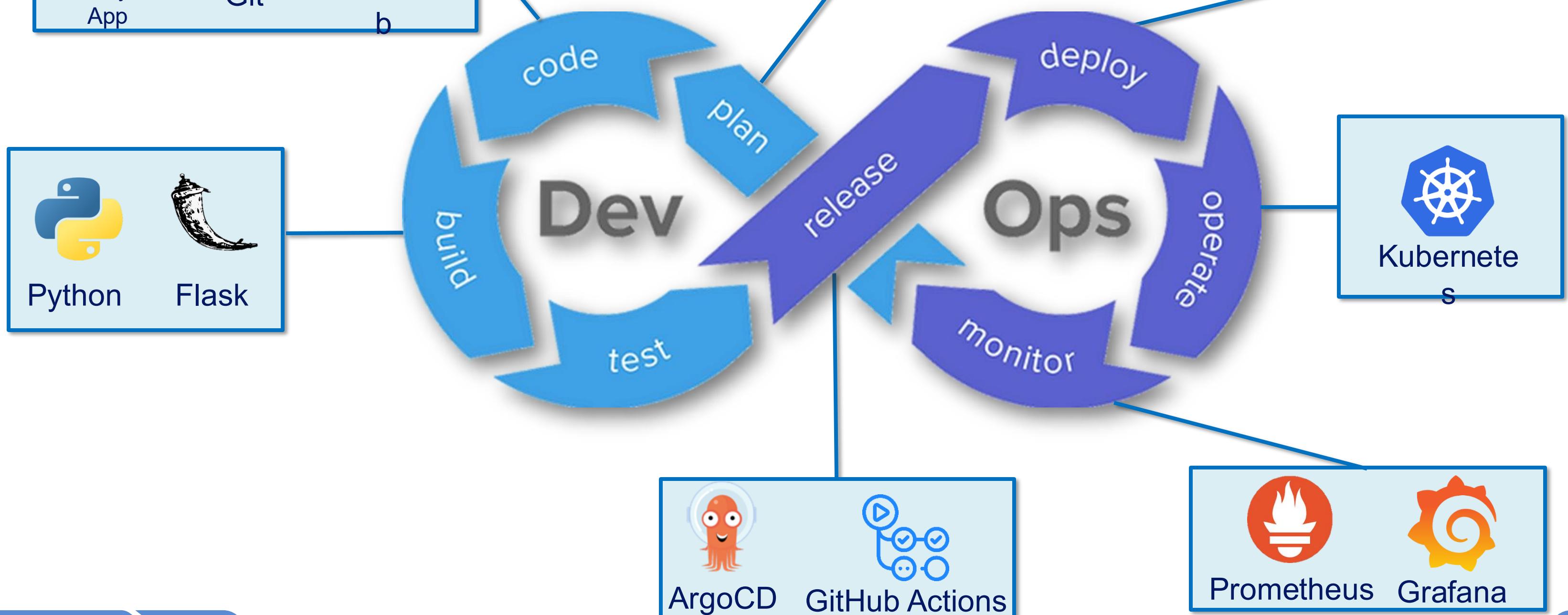
Setting up your Development Environment



Summary

- ❖ Explored DevOps, its evolution, and benefits
- ❖ Understood the DevOps lifecycle and how each stage fits together
- ❖ Introduced CI/CD and Continuous Operations

Summary



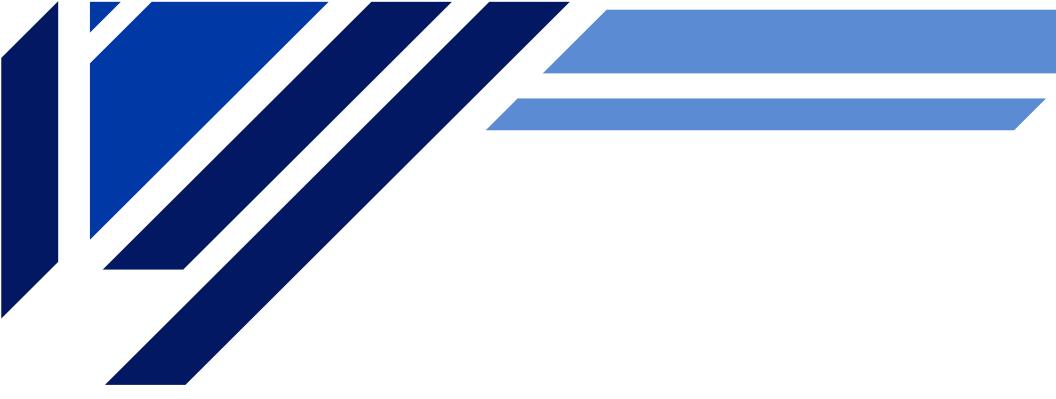


THANK YOU

Section 2

Planning with GitHub Projects & Issues

Presented By: Thinknyx



Agile Workflows

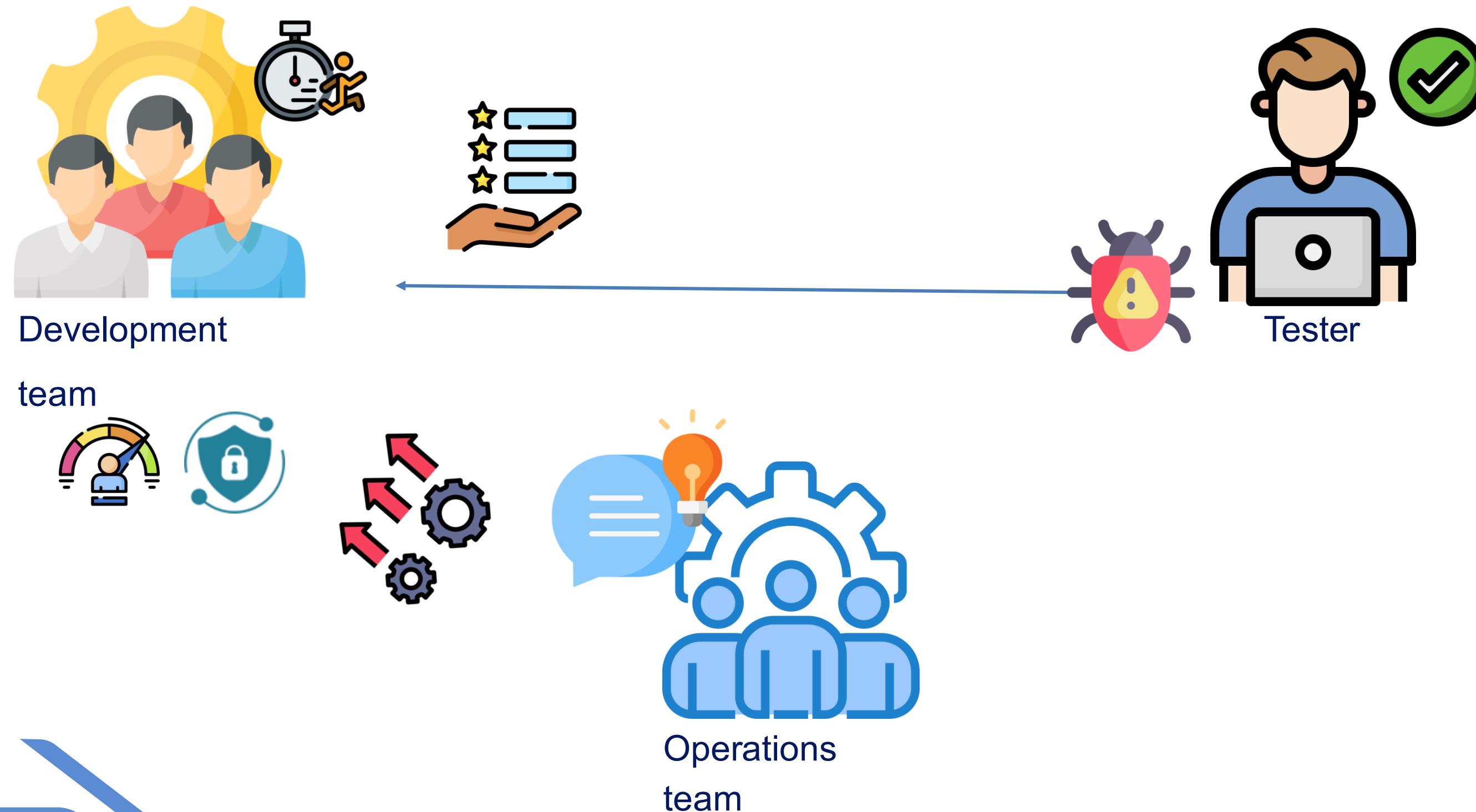


Agile Workflows

DevOps = Collaboration + Automation

Agile Workflows

➤ The Challenge of Visibility



Agile Workflows

➤ The Challenge of Visibility



What to prioritize first?

- Fix critical bug?
- Continue with new feature?
- Handle feedback?
- Confusion for aligning across **developers, testers, and operations**

Agile Workflows

➤ The Role of Project Management



Large
Organizations



Project
manager



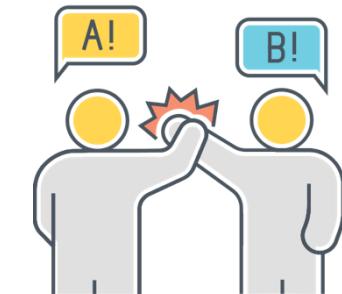
Smaller teams



Manager

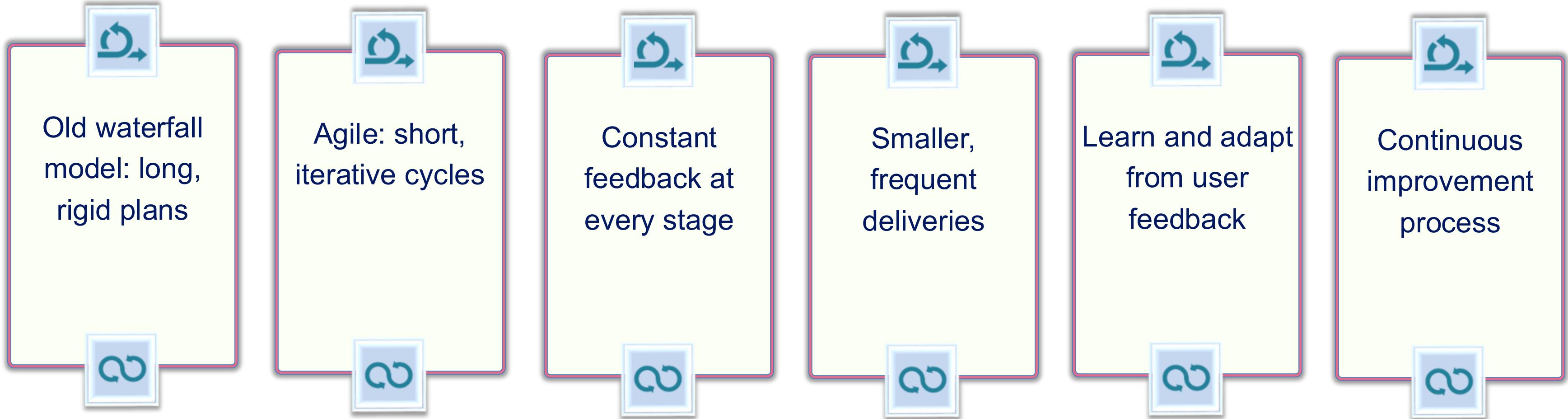


- Without project management, CI/CD pipelines can become chaotic
- Automation alone is not enough
- Visibility is essential for smooth operations
- Coordination ensures teams work in sync



Agile Workflows

➤ Agile and DevOps Alignment



Agile Workflows

➤ Agile Workflows: Scrum and Kanban

Scrum boards

Kanban boards

What's in the backlog

what's currently being worked on

what's already done

No confusion

Entire team stays aligned

Agile Workflows

➤ Scrum

- Work organized into short, focused sprints
- Team commits to a set of tasks at sprint start
- Review results and gather feedback at sprint end
- Provides structure and predictability
- Ideal for teams wanting clear goals and regular checkpoints



Two
weeks

Agile Workflows

➤ Kanban

- Kanban emphasizes continuous flow, not sprints
- Work items pass through stages: To Do → In Progress → Testing → Done
- Focus on smooth task movement across stages
- Apply Work-In-Progress (WIP) limits to avoid overload
- Identify and remove bottlenecks quickly



Agile Workflows

➤ Agile Workflows: Scrum and Kanban

Scrum boards

- Structured approach
- Time-boxed sprints

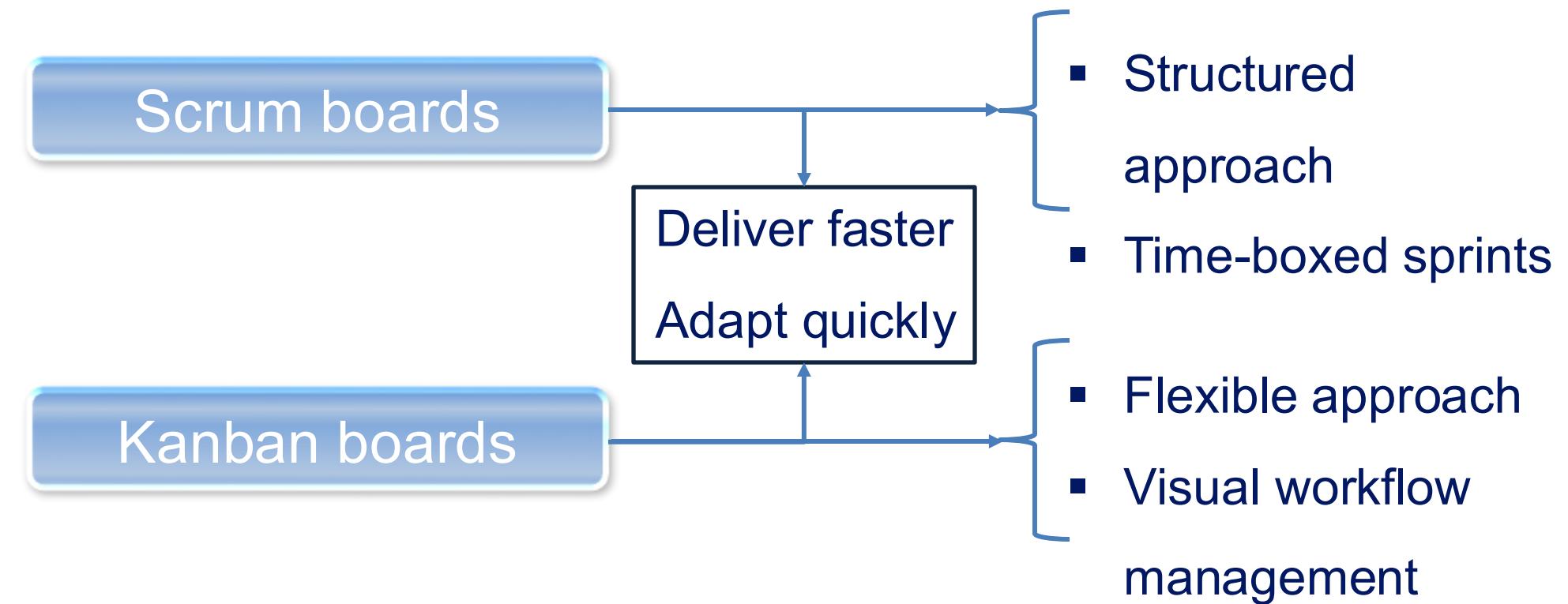
Kanban boards

- Flexible approach
- Visual workflow management

- ✓ Deliver faster
- ✓ Adapt quickly

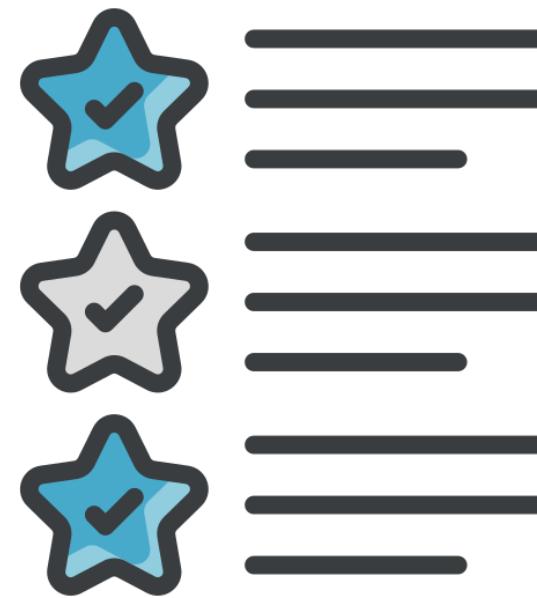
Agile Workflows

➤ Agile Workflows: Scrum and Kanban



Agile Workflows

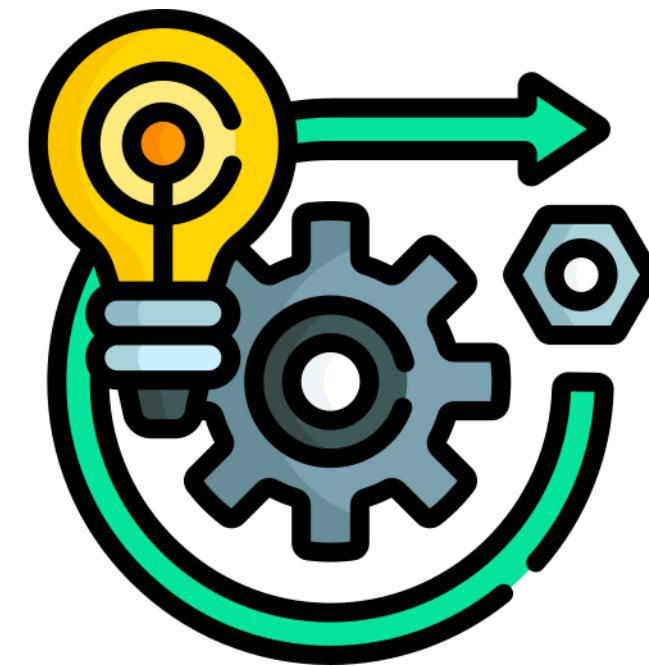
➤ Agile Workflows: Scrum and Kanban



Features



Bugs



Operations

Agile Workflows

➤ Agile Workflows: Scrum and Kanban



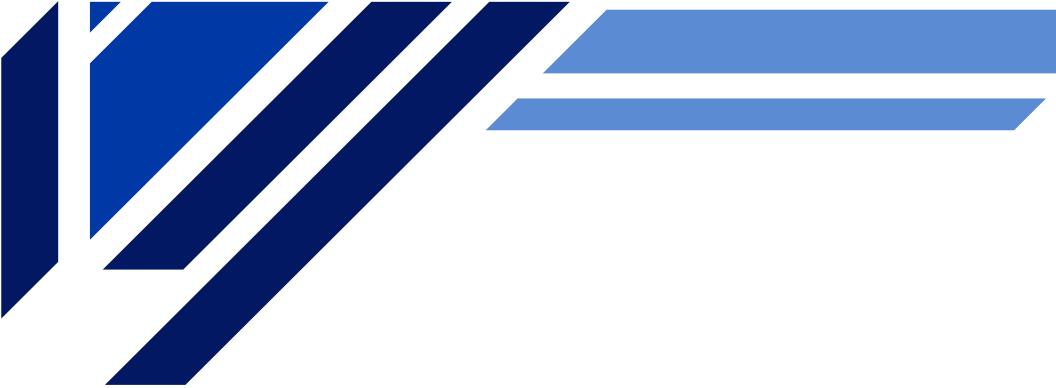
- Priorities are Transparent – Everyone knows what matters most
- Progress is Visible – Clear tracking of tasks and deliverables
- Collaboration is Smoother – Teams stay aligned and coordinated

Agile Workflows

➤ Modern Tools for Agile in DevOps



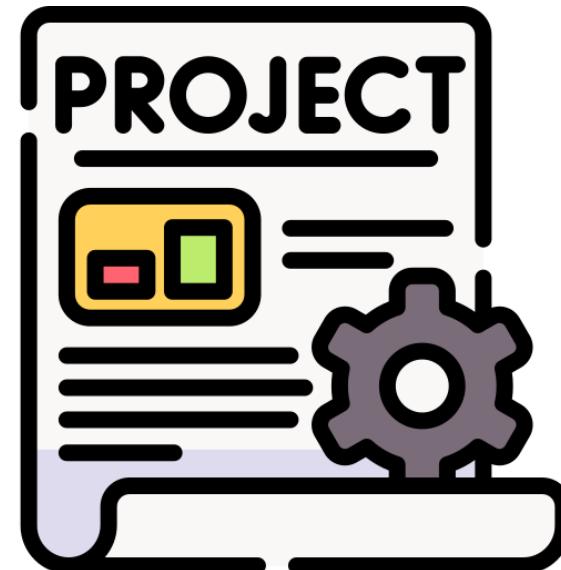
- Shift from sticky notes to digital boards
- Accessible from anywhere
- GitHub Projects as a powerful tool
- Integrates Scrum and Kanban
- Works within the same platform where code already lives



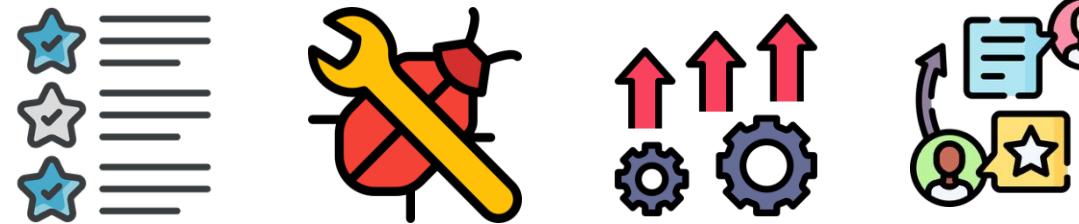
GitHub Projects and Issues



What is a Project?



Purpose: to create a unique product, service, or result



❑ Project management

tools



Jira



Trello

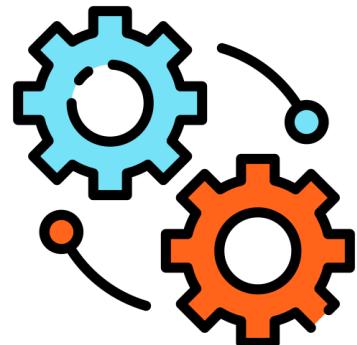


Asana

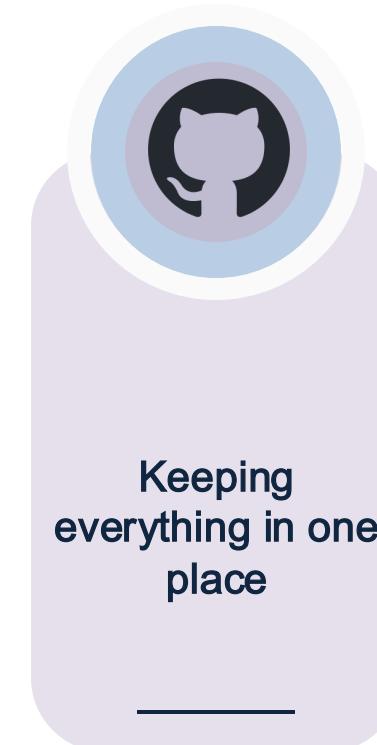


GitHub Projects

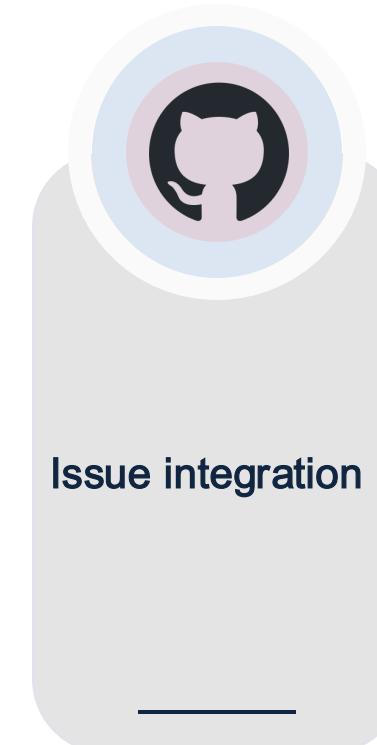
Why GitHub Projects?



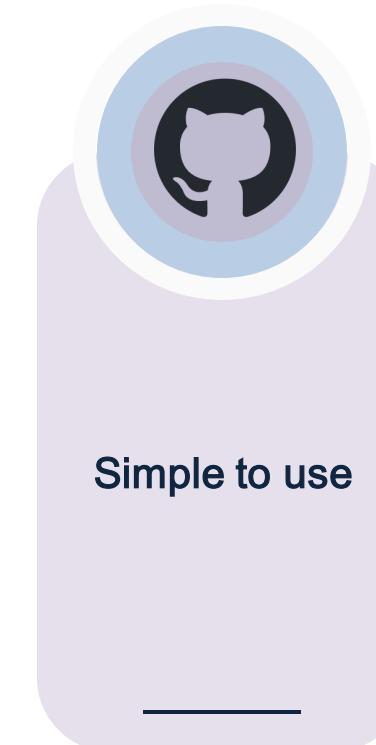
Integration



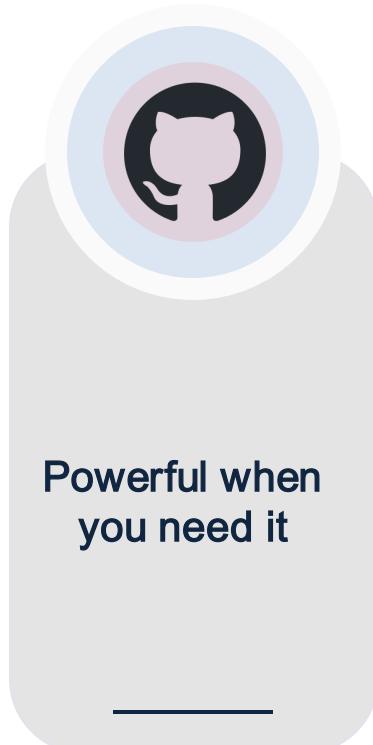
Keeping
everything in one
place



Issue integration



Simple to use



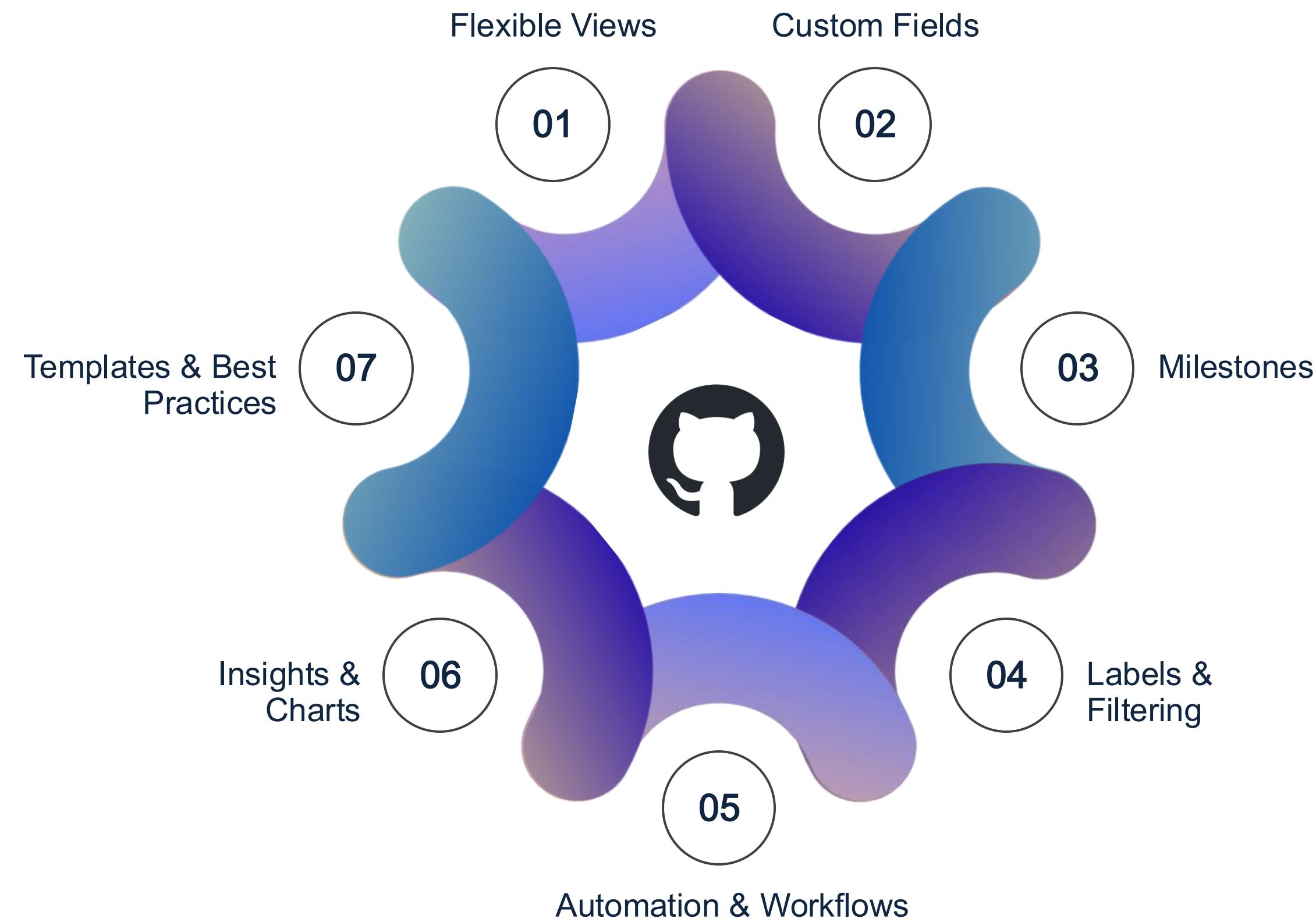
Powerful when
you need it

GitHub Issues: The Foundation of Work



- Track bug reports, feature requests, tasks, or ideas with Issues
- Create issues from repository, PR comment, code line, GitHub Desktop, CLI, or mobile app
- Break down issues into sub-issues for smaller, trackable tasks

Features of GitHub Projects



Summary

- ❖ GitHub Issues: Capture ideas, bugs, and tasks
- ❖ GitHub Projects: Organize, prioritize, and track work
- ❖ Combined Power: Lightweight yet powerful project management system integrated with your code

Demo

DevOps | Creating Your GitHub Account

Demo

DevOps | Set up a GitHub Project board



THANK YOU

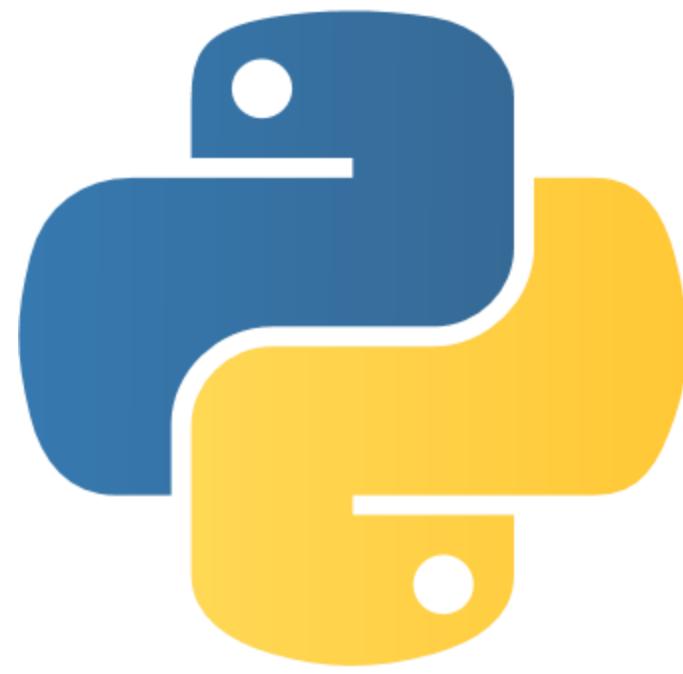
Section 3

Python Application Fundamentals

Presented By: Thinknyx

Introduction to Python

Introduction to Python



Python



Flask

Introduction to Python



DevOps Integration

Running applications through DevOps pipeline



Application Building

Using Python to create applications

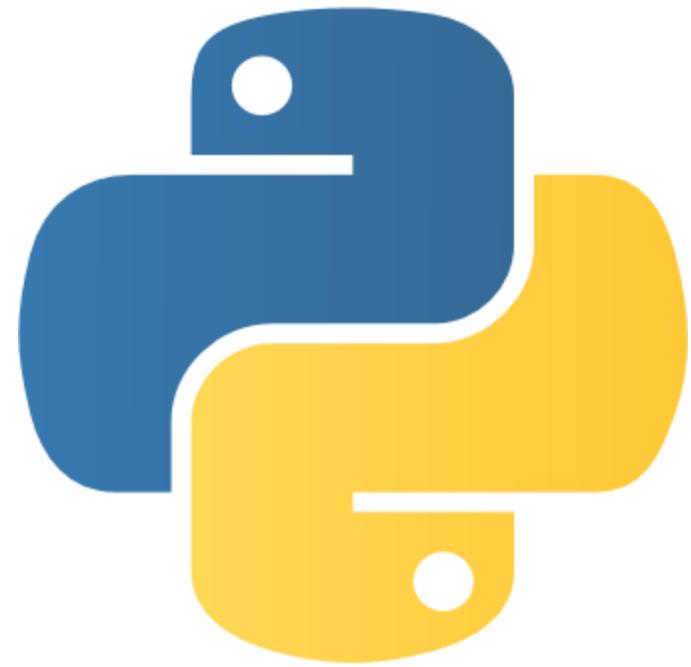


Python Essentials

Basic Python concepts for beginners

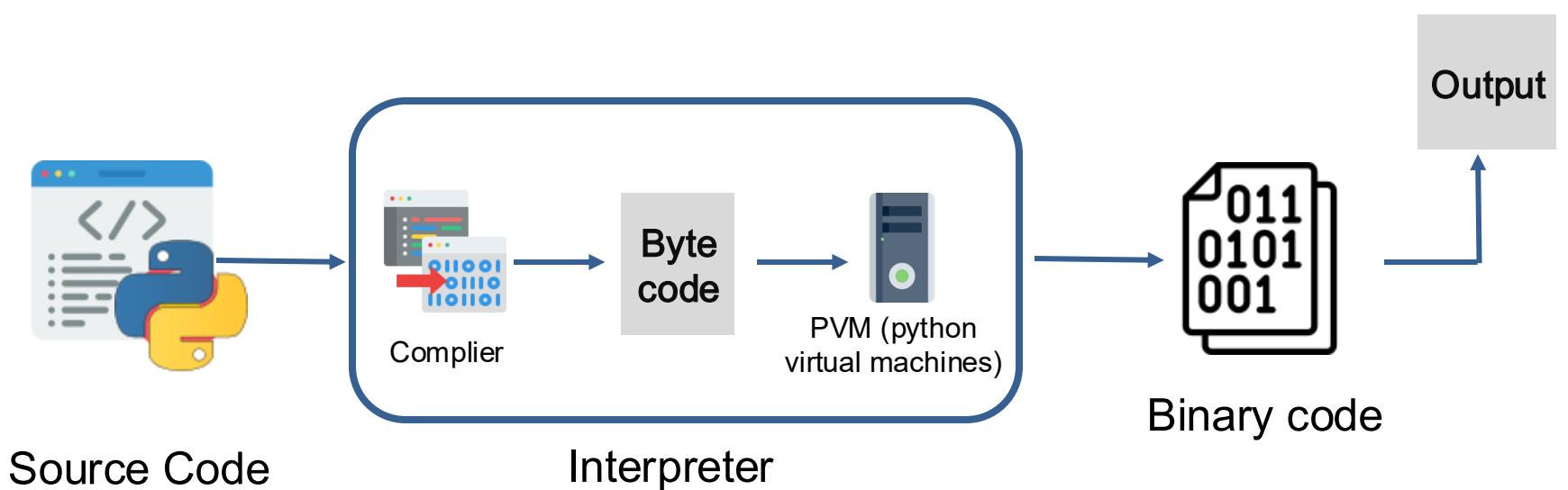
What is Python?

- High-level, easy-to-learn language
- Clear and simple syntax
- Suitable for both large systems and quick scripts



What is Python?

- High-level, easy-to-learn language
- Clear and simple syntax
- Suitable for both large systems and quick scripts
- **Interpreted** – run code directly without compiling
- Faster and interactive development
- Freely available on all major platforms
- Huge standard library
- Large ecosystem of third-party modules
- One of the most versatile programming languages



Key Features of Python



Easy to Understand



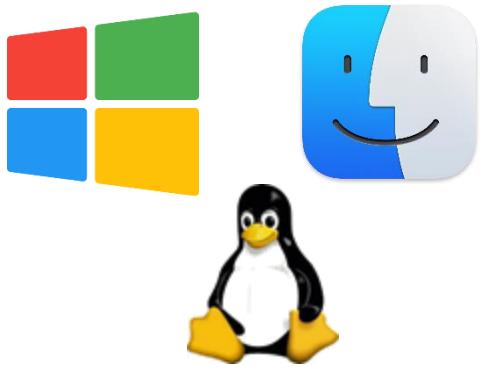
Object-Oriented Programming



Easy Integration



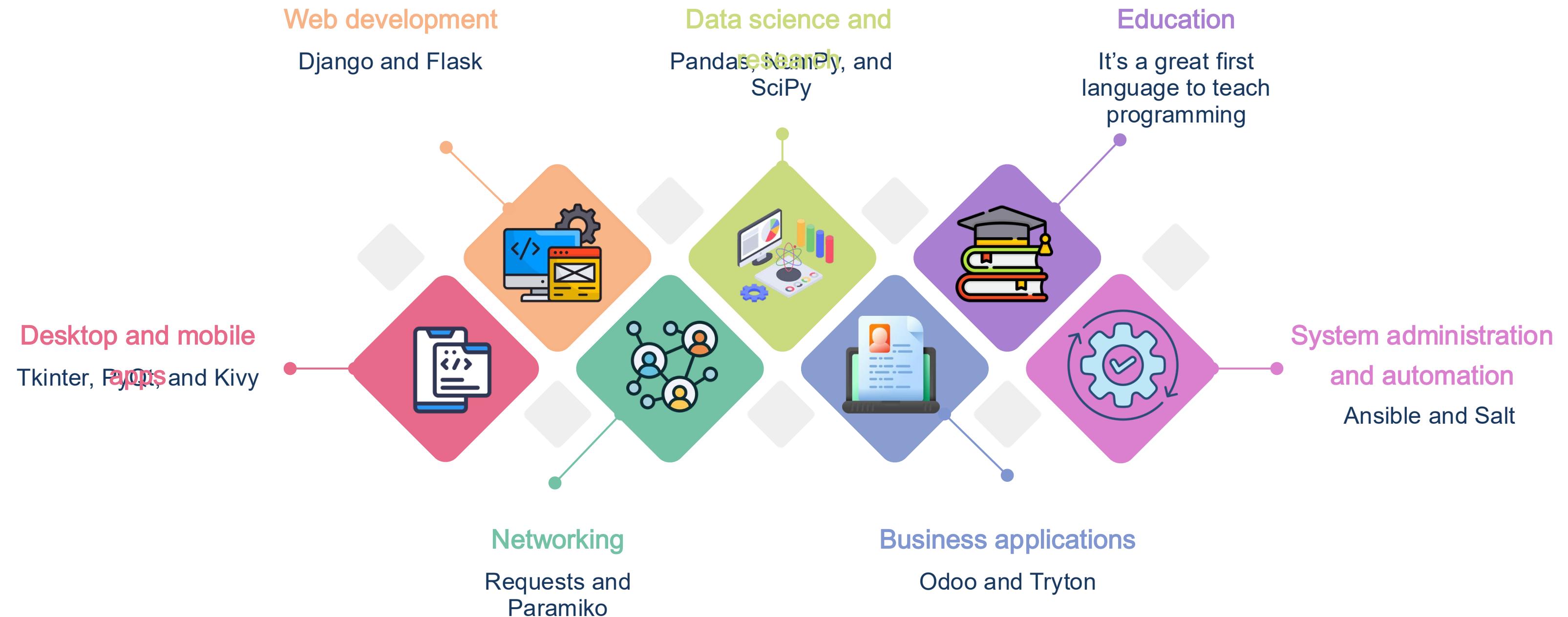
Works Across Platforms



Supportive Community



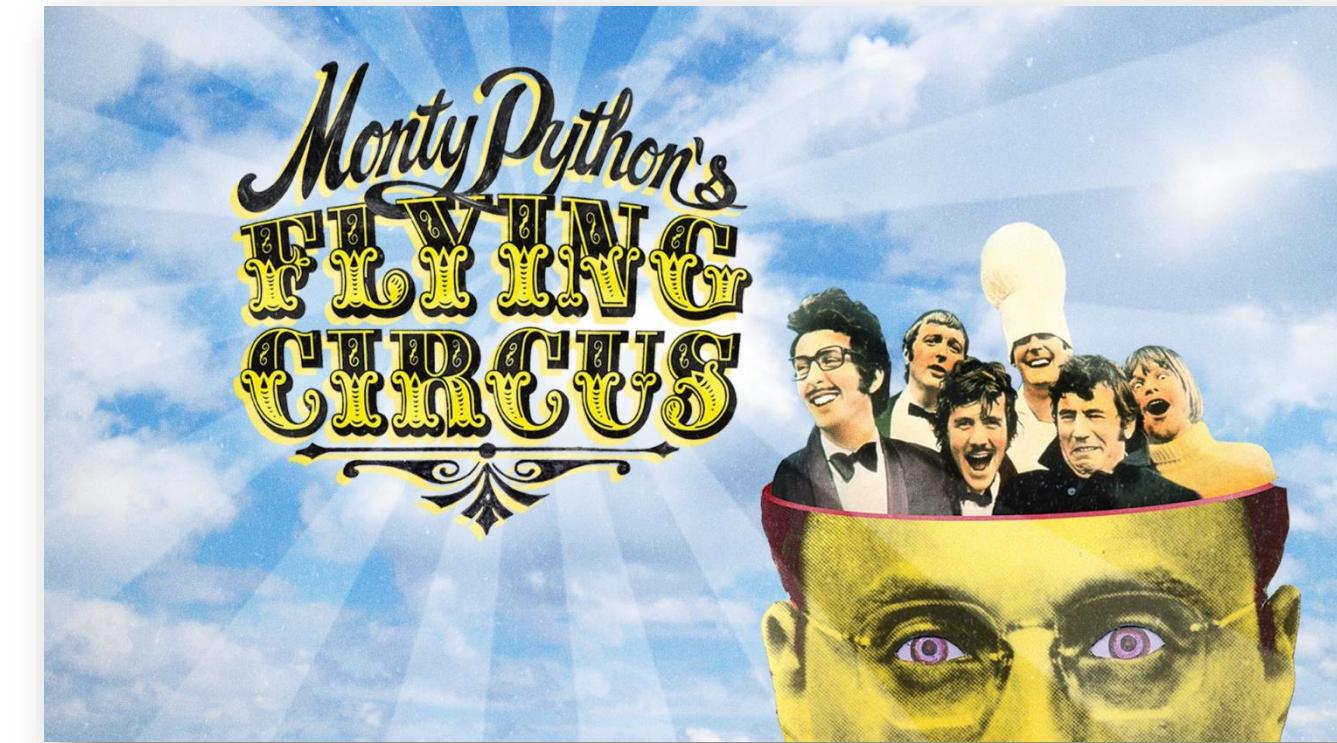
Use Cases of Python



Who Started Python?

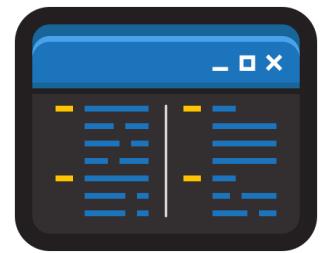


Guido van Rossum
1990's



Getting Started with Python Basics

Python Files

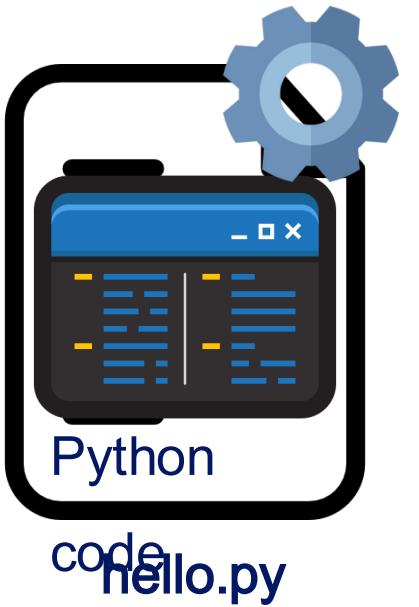


Python
code

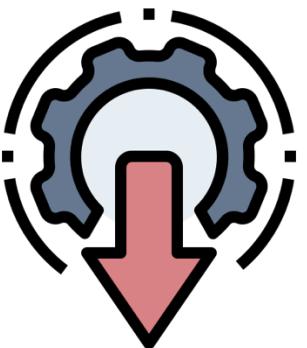


.py

Python Files

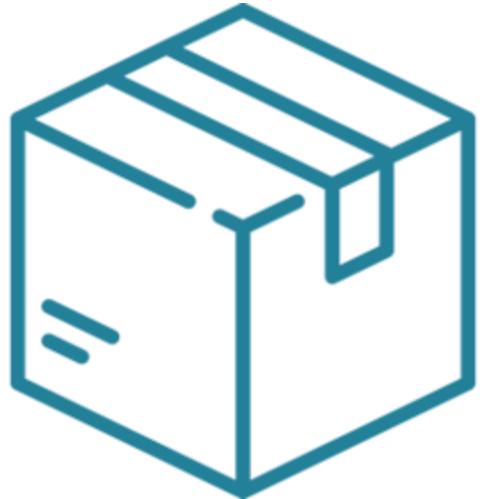


```
print("Hello, World!")
```



Hello, World!

Variables



```
organization = "Thinknyx"
```

Reserved keywords

- if
- for
- class

Data Types

Numbers

- **Example**
 - `x = 10` is an integer
 - `y = 3.5` is a decimal number

Strings

- **Example**
 - `name = "Thinknyx"`

Booleans

- **Example**
 - `is_active = True`

Arithmetic Operators

“Plus” sign for addition: $10 + 5$ gives 15

“Minus” sign for subtraction: $10 - 5$ gives 5

“Star” for multiplication: $10 * 5$ gives 50

“Slash” for division: $10 / 5$ gives 2

Control Flow

If-Else

```
organization = "Thinknyx"
age = 18
if age >= 18:
    print("You are eligible to vote.")
else:
    print("You are not eligible.")
```

while loop

```
count = 0
while count < 3:
    print("Welcome!")
    count += 1
```

Loops

```
for i in range(3):
    print("Thinknyx")
```

Functions

- Define a function

```
def greet(name):  
    return f"Hello, {name}!"
```

Parameter

- Call the function

```
message = greet("Thinknyx")  
print(message)
```

Argument

Modules

- A module is just another **Python file** that contains useful code
 - Example

```
import math  
print(math.sqrt(16))
```



Object and Class

- A class as a
blueprint What something is
 - What it can do

```
class Dog:  
    def bark(self):  
        print("The dog is barking!")
```

- An object is the actual thing created from that plan

```
my_dog = Dog("Tommy")  
my_dog.bark()
```

Indentation

- Example

```
if True:  
    print("This is indented correctly")
```

```
if True:  
print("This will fail")
```



- Indent blocks inside **if-else**, **loops**, and **functions**
- Keep indentation consistent in code
- Standard practice: 4 spaces per level

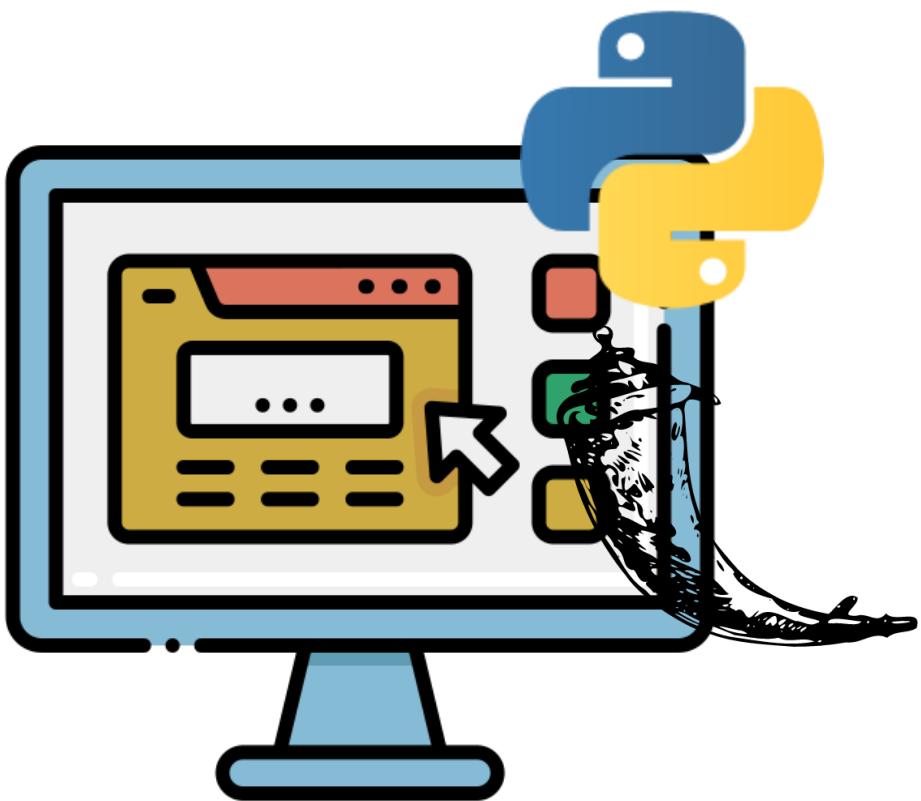
Demo

DevOps | Python in Action

Demo

DevOps | Create a sample Python Flask app

Create a sample Python Flask app



What is Flask?

- Lightweight web framework in Python
- Helps you start quickly and scale easily
- Provides simplicity + flexibility
- Very popular among Python developers
- Consistently ranked top framework (2018–2022 surveys)
- Strong community support
- Beginner-friendly yet production-ready



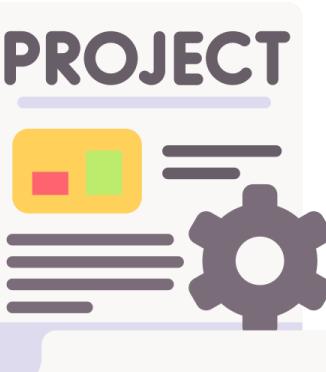
Why a Virtual Environment

- A virtual environment is an isolated workspace for one project



Flask 2.x

Specific Jinja version



Flask 3.x

Newer Jinja



Summary

- ❖ Python applications don't require a build step
- ❖ Unlike Java or C#, no compilation with tools like **Maven**, **Gradle**, or **Visual Studio**
- ❖ No separate test execution needed before running (**JUnit**, **NUnit**, etc.)
- ❖ With **Python + Flask**: just run the app and open in browser
- ❖ Direct testing makes development faster and more interactive



Unlocking Python for the Absolute Beginners – Hands-on



THANK YOU

Section 4

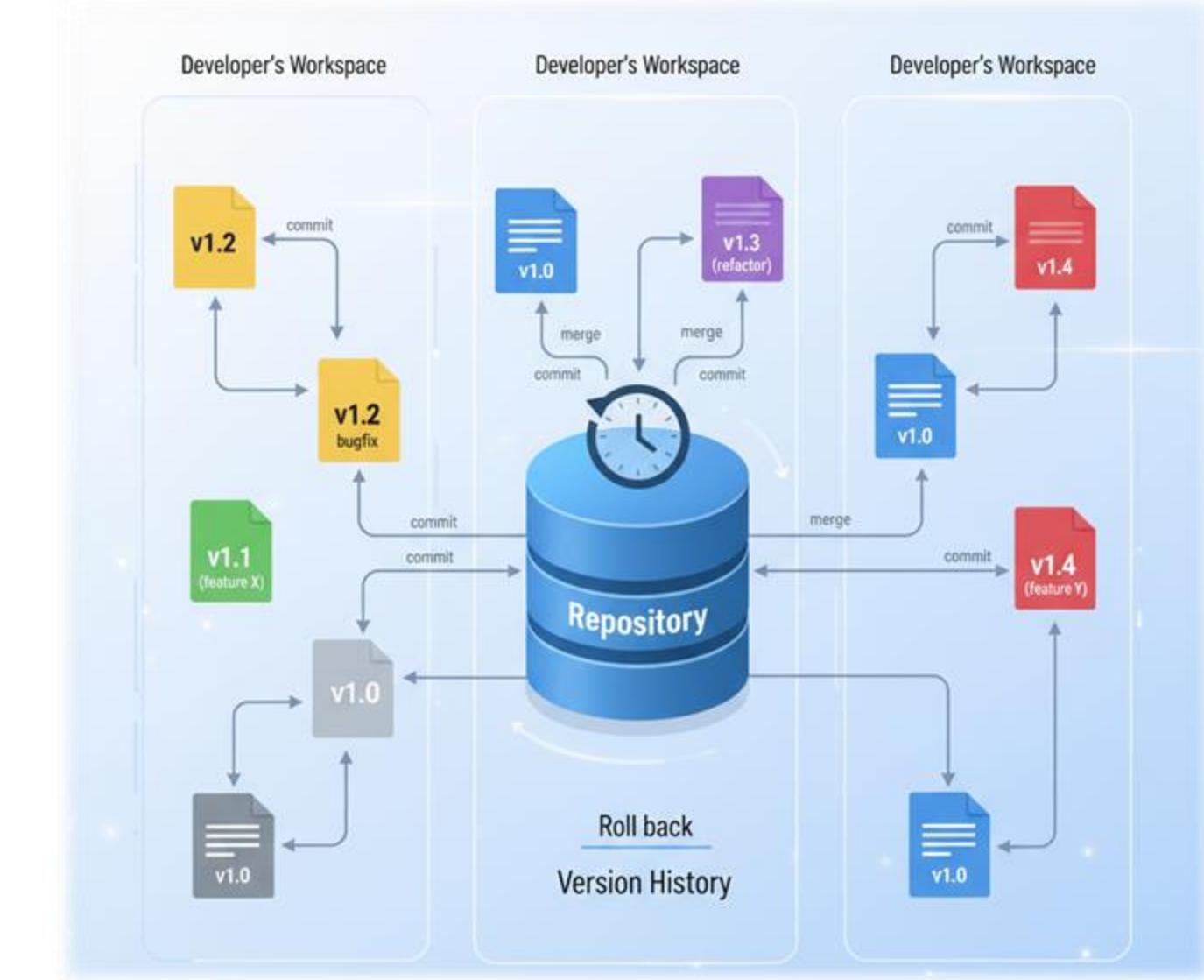
Version Control with Git and GitHub

Presented By: Thinknyx

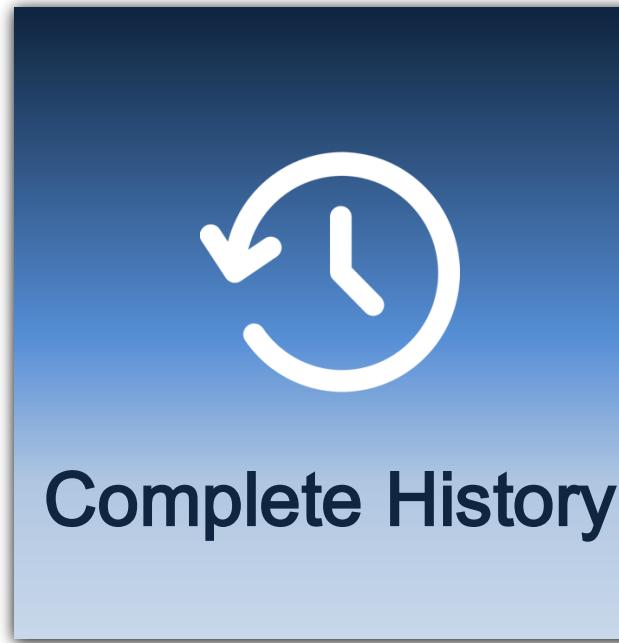
Version Control with Git and GitHub

What is a Version Control System (VCS)?

- Tracks and manages code changes
- Automatically records every change in a special database
- Allows reviewing older versions of code
- Enables comparing changes between versions
- Provides rollback to a stable state if issues occur



Why Use a Version Control System?



Complete History



Branching and
Merging



Traceability

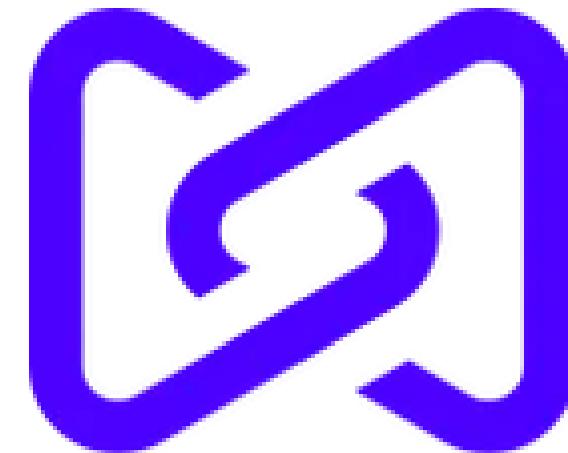
Different Version Control Tools



Subversion



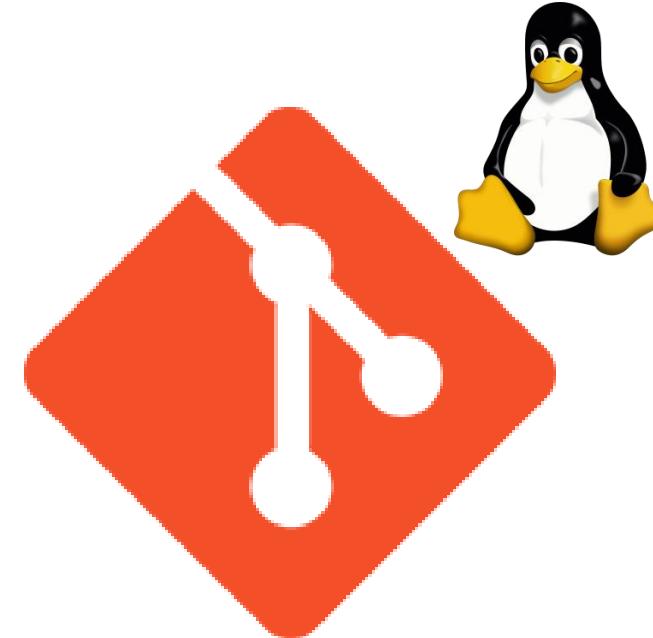
Mercurial



Perforce



Git



Distributed version control system



Linus Torvalds

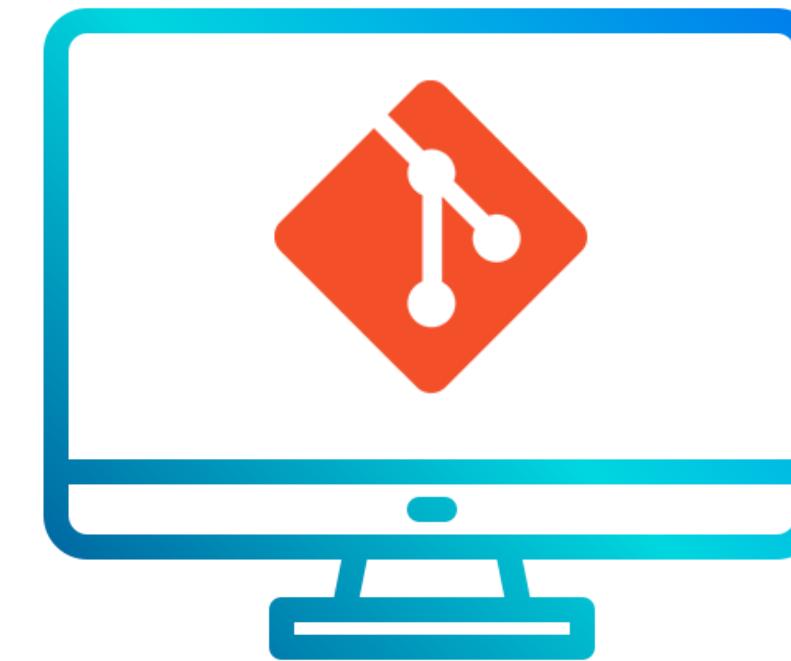
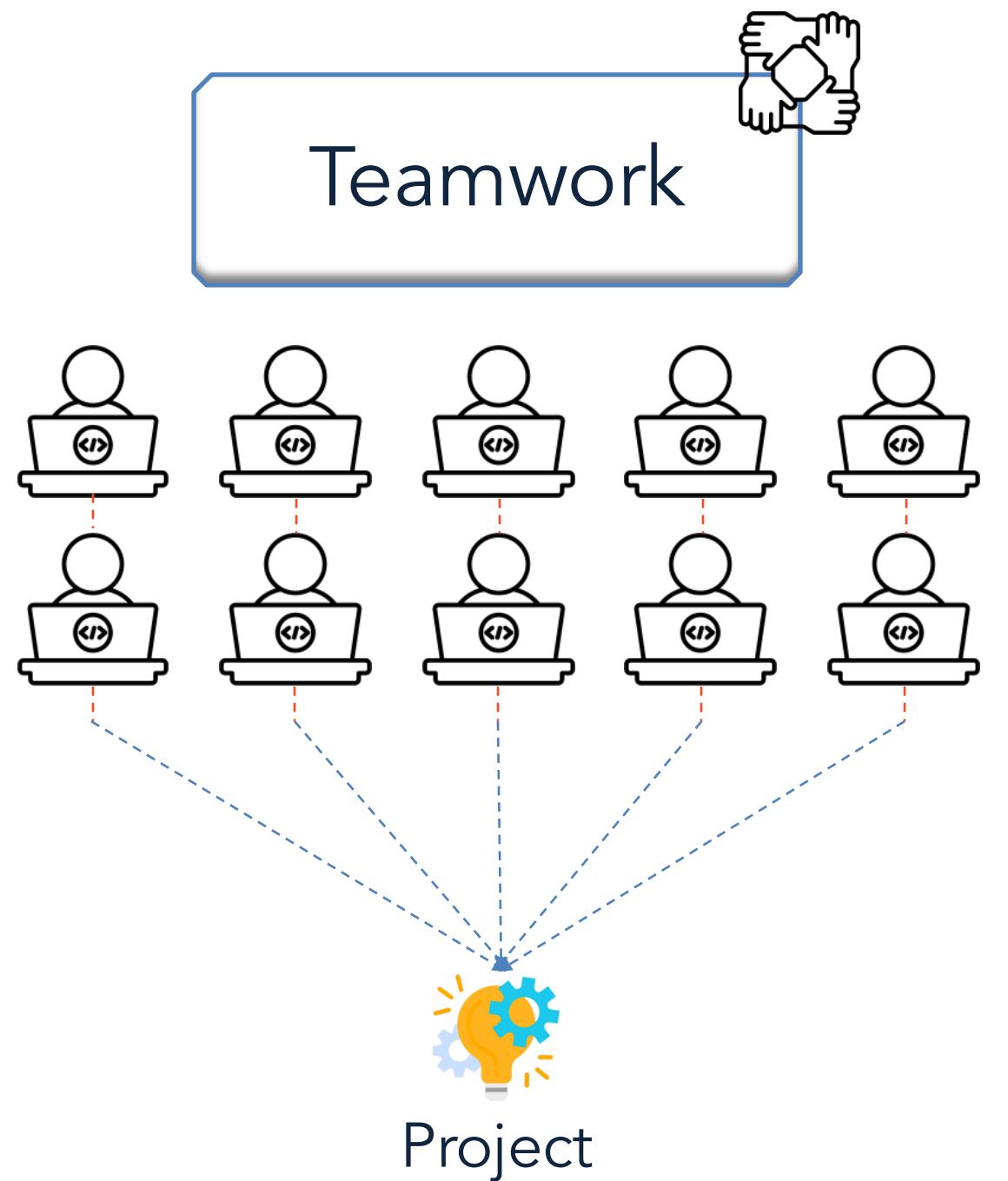


Junio C Hamano

2005

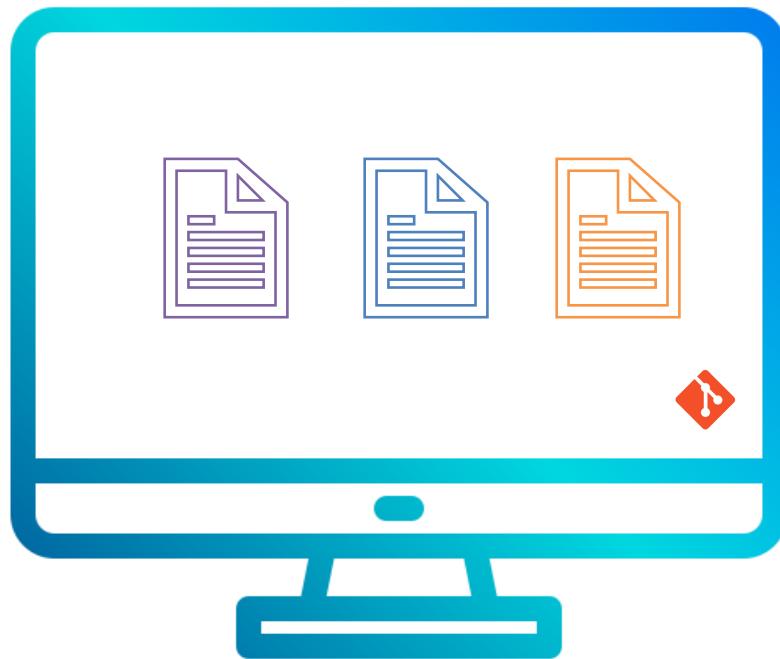
- Built for Linux community's need: **fast, reliable, distributed VCS**
- Became the most widely used Version Control System (VCS)
- Currently maintained by Junio C. Hamano (lead developer for many years)

Git vs GitHub



- Use Git to track code changes
- Maintains full history of changes
- Roll back to previous versions easily
- Create branches for experiments and new features

Git vs GitHub



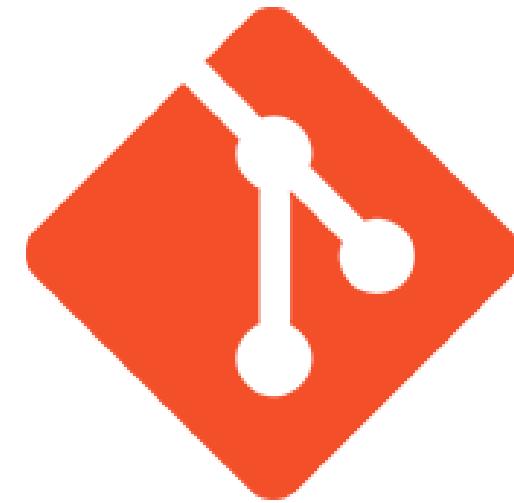
- How do you share your code with your teammate easily?
- How do you both collaborate without emailing files back and forth?

Git vs GitHub

- Online platform built on Git
- Upload repositories to the cloud
- Collaborate with team members from anywhere
- Review and manage code together
- Automate workflows for efficiency



Git vs GitHub



Local version control
tool

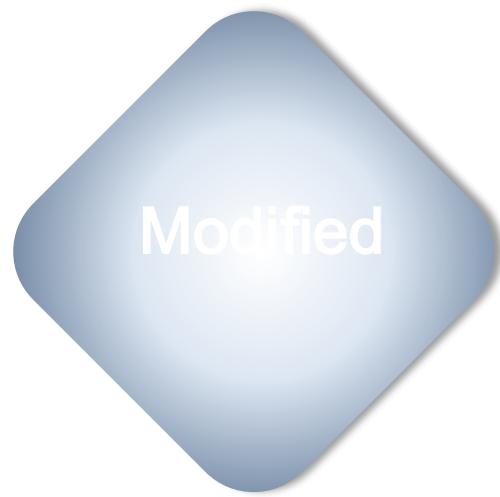


Online collaboration
platform

Git Workflows

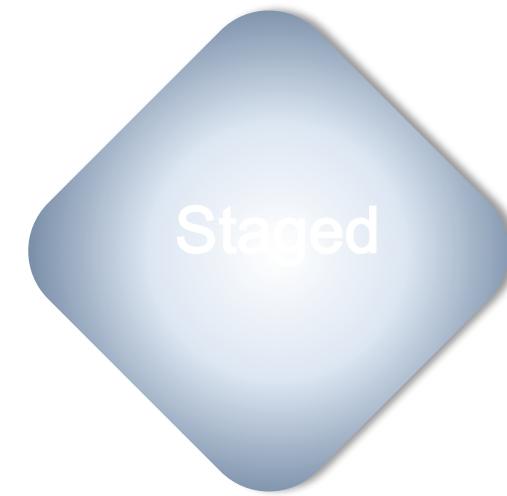
Git Workflows

➤ Three States in Git



Modified

Changes made to file,
not yet saved in Git
history (check with `git
status`)



Staged

File marked for next
commit using `git add
filename` or `git add .`

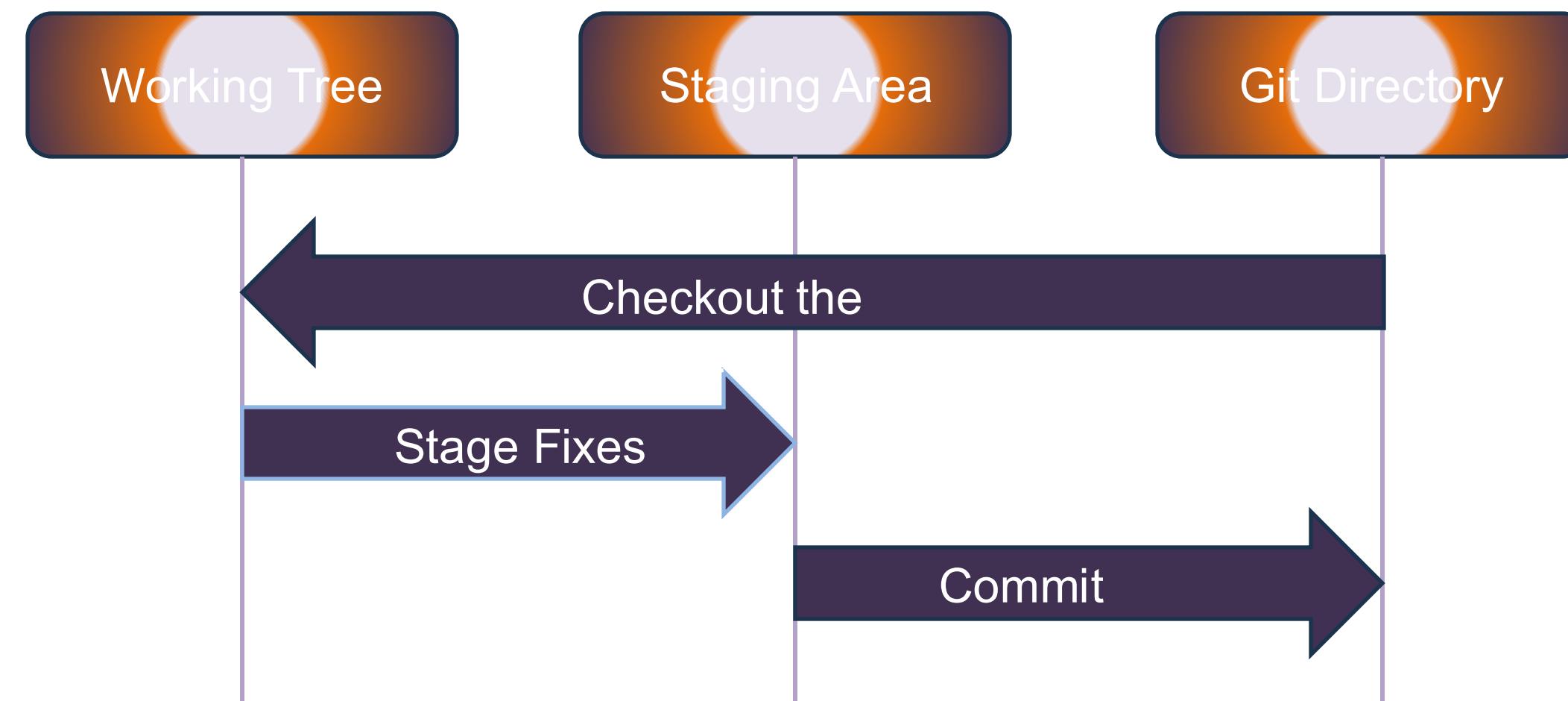


Committed

Changes permanently
saved in Git's local
database with `git commit
-m "message"`

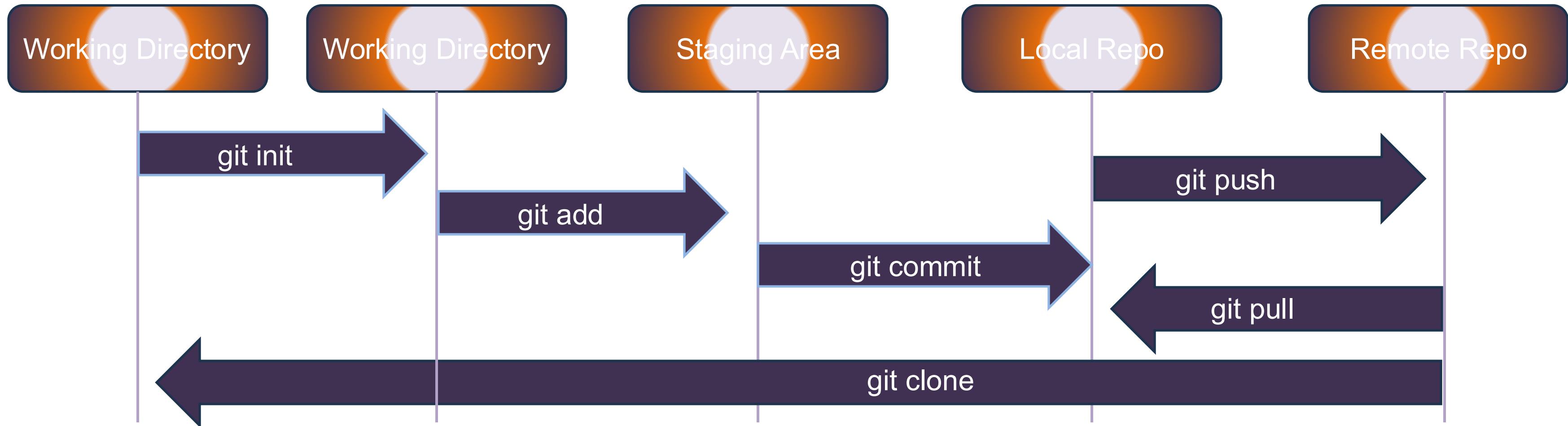
Git Workflows

➤ The Three Main Sections



Git Workflows

➤ The Basic Workflow



Demo

DevOps | Push the app to GitHub

Demo

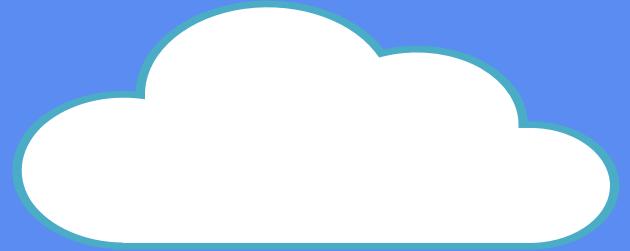
DevOps | Managing Course Development with GitHub Projects



THANK YOU

Cloud Infrastructure with AWS

Section 5



Setting Up Real-World Infrastructure with AWS

Setting Up Real-World Infrastructure with AWS

Orchestration

Monitoring

Continuous operations

Automating CI/CD



Containerize the
application

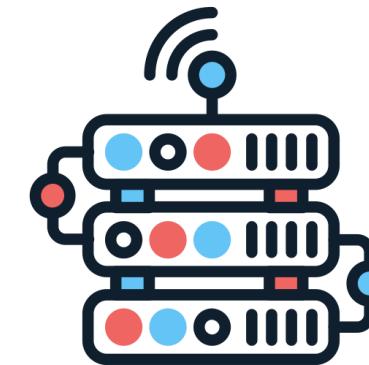


Center

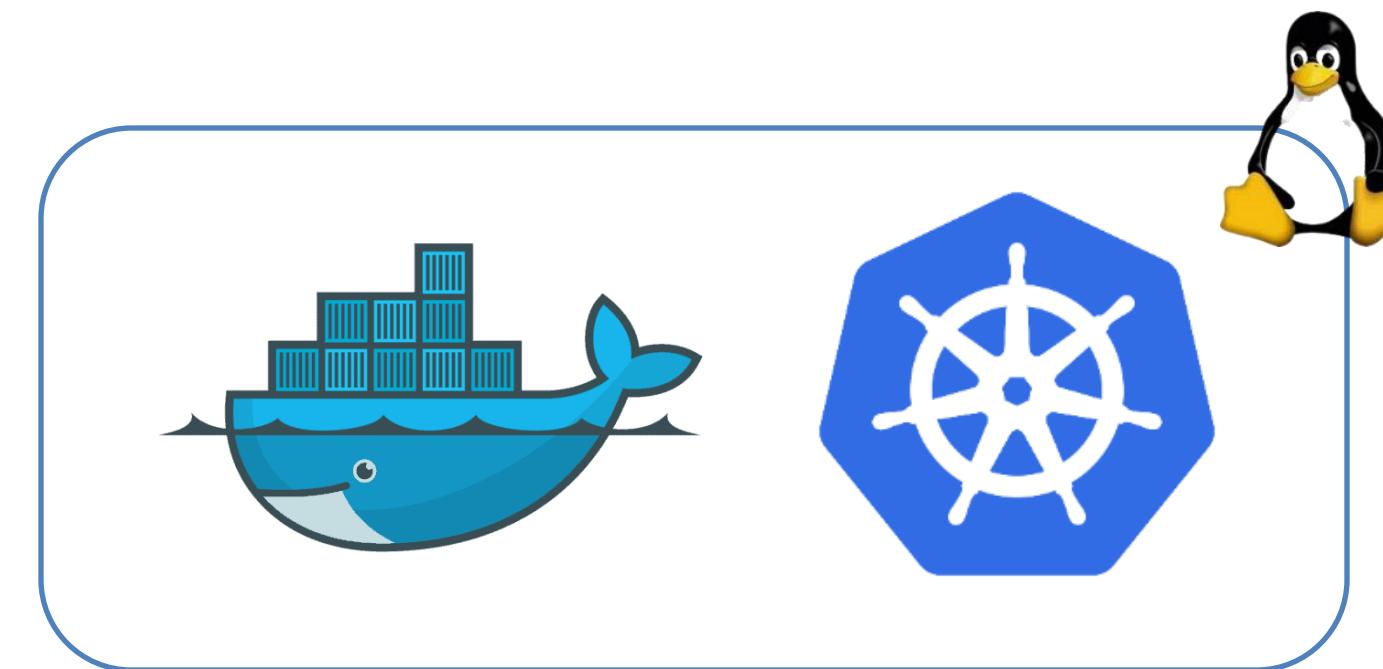
Setting Up Real-World Infrastructure with AWS



Pay only for what you use
which makes EC2 a cost-effective and flexible



Setting Up Real-World Infrastructure with AWS



- **Problem:** Running Linux locally is resource-intensive on your machine
- **Solution:** We'll work directly on Linux running inside EC2 servers

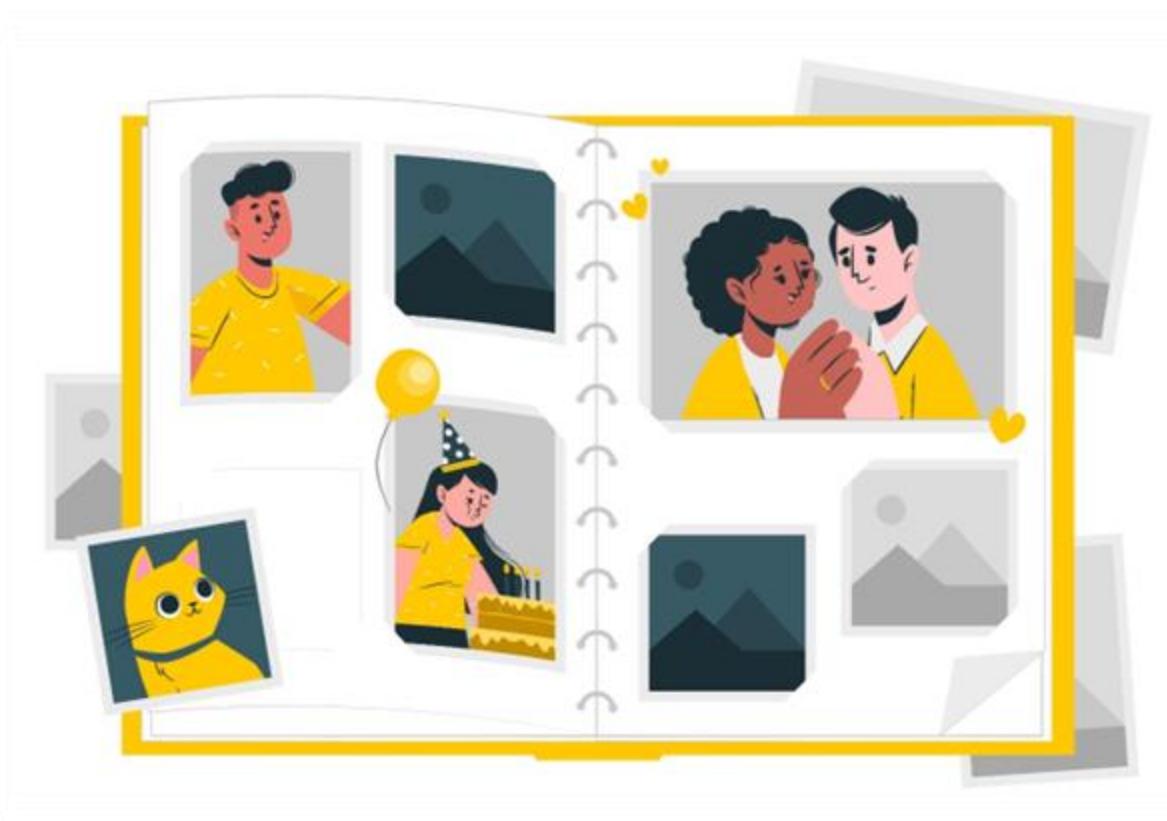
Setting Up Real-World Infrastructure with AWS

- ❖ Basics of cloud computing
- ❖ Introduction to AWS
- ❖ Creating an AWS account
- ❖ Best practices for launching an EC2 instance

Introduction to Cloud

Introduction to Cloud

➤ What is Cloud Computing?



25 to 30 Pictures

Introduction to Cloud

➤ What is Cloud Computing?



Introduction to Cloud

➤ What is Cloud Computing?



GooglePhotos



Google Drive



OneDrive



iCloud

- ✓ Even if you lose your device, your files stay safe in the cloud

Introduction to Cloud

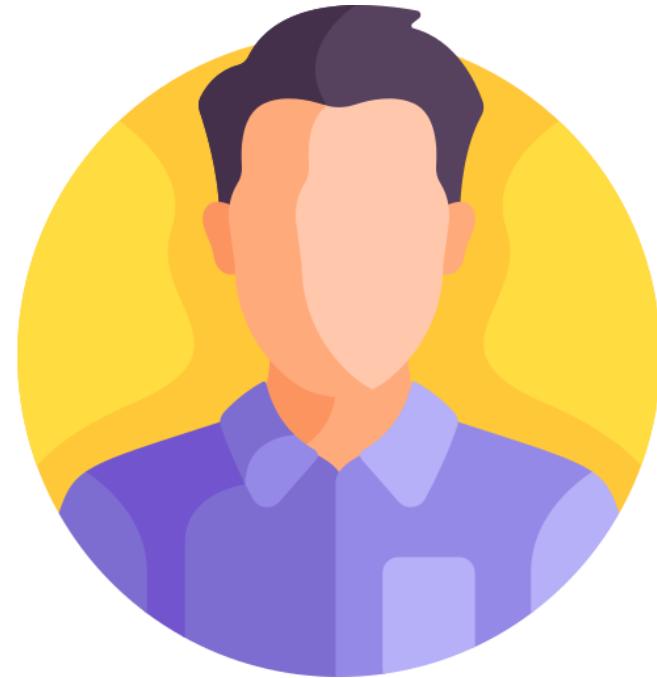
➤ What is Cloud Computing?

- ✓ Access your data anytime, anywhere
- ✓ Works across devices with an internet connection
- ✓ Seamless switching between devices
- ✓ Ideal for remote work and streaming



Introduction to Cloud

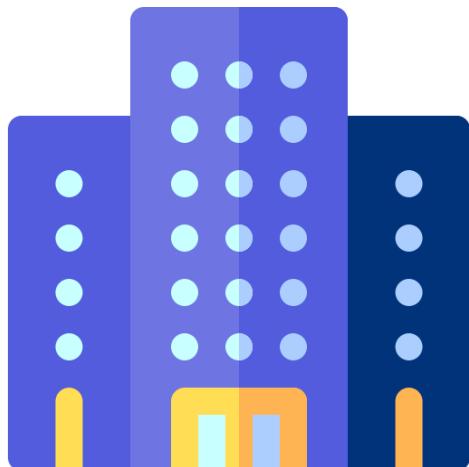
➤ What is On-Premises Computing?



Introduction to Cloud

➤ What is On-Premises Computing?

- ✓ **Traditional IT Model** – Organization owns, manages, and maintains hardware and software
- ✓ **Legacy Systems** – Supports mainframes, Solaris, HP-UX, and other enterprise technologies
- ✓ **Physical Infrastructure** – Servers, storage, and networking housed in dedicated data centers or server rooms



- ✓ **Humble Beginnings** – Many businesses start small, just like Google began in a garage
- ✓ **Traditional Approach** – Initial reliance on minimal infrastructure before growing
- ✓ **Gradual Expansion** – Organizations scale up, investing in their own data centers

Introduction to Cloud

➤ Limitations of On-Premises Computing

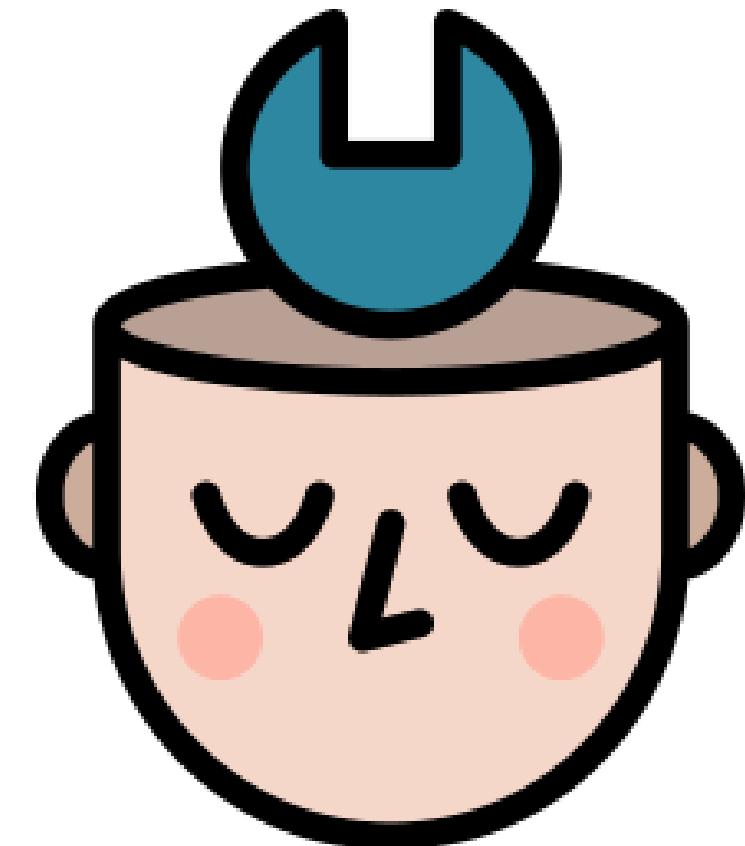
- **High Costs & Upfront Investments** – Requires significant capital for hardware, storage, and infrastructure, with risky resource planning



Introduction to Cloud

➤ Limitations of On-Premises Computing

- **Complex Maintenance & IT Overhead** – Needs a dedicated IT team for upkeep, increasing costs and operational complexity



Introduction to Cloud

➤ Limitations of On-Premises Computing

- **Scalability Challenges**

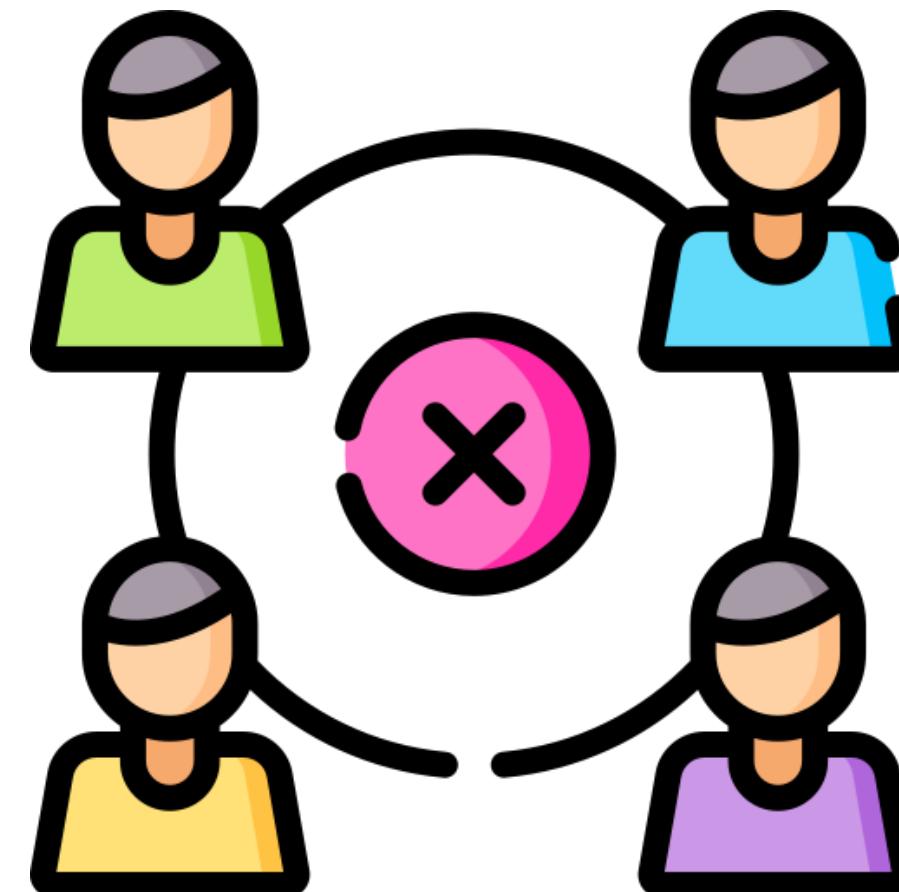
- **Overprovisioning** – Excess capacity leads to wasted resources
- **Underprovisioning** – Insufficient capacity causes performance issues and downtime



Introduction to Cloud

➤ Limitations of On-Premises Computing

- **Limited Accessibility & Collaboration** – Restricted remote access hinders global teamwork



Introduction to Cloud

➤ Limitations of On-Premises Computing

- **Backup & Disaster Recovery Challenges** – Managing backups and recovery solutions is costly and complex, increasing the risk of data loss



Introduction to Cloud

➤ Why Cloud Computing?

- More flexible
- Scalable
- Cost-effective

Introduction to Cloud

➤ Advantages of moving to the cloud

✓ Cost Efficiency: Pay for What You Use



No Upfront Investment – Avoid large capital expenses on hardware

Pay-as-You-Go – Only pay for the resources consumed

On-Demand Scaling – Increase or decrease capacity based on need

Avoid Overprovisioning – Prevent unnecessary spending on idle resources

Example: Black Friday – Temporarily scale up servers and scale down post-event

Optimized Costs – Reduce waste and improve financial efficiency

Introduction to Cloud

➤ Advantages of moving to the cloud

✓ Simplified Maintenance & Reduced IT Overhead

Managed Infrastructure – Cloud providers handle hardware, updates, and security

Less IT Burden – No need for large in-house IT teams for maintenance

Automatic Updates – Software and security patches are handled by the provider

Focus on Innovation – IT teams can prioritize business growth instead of server management

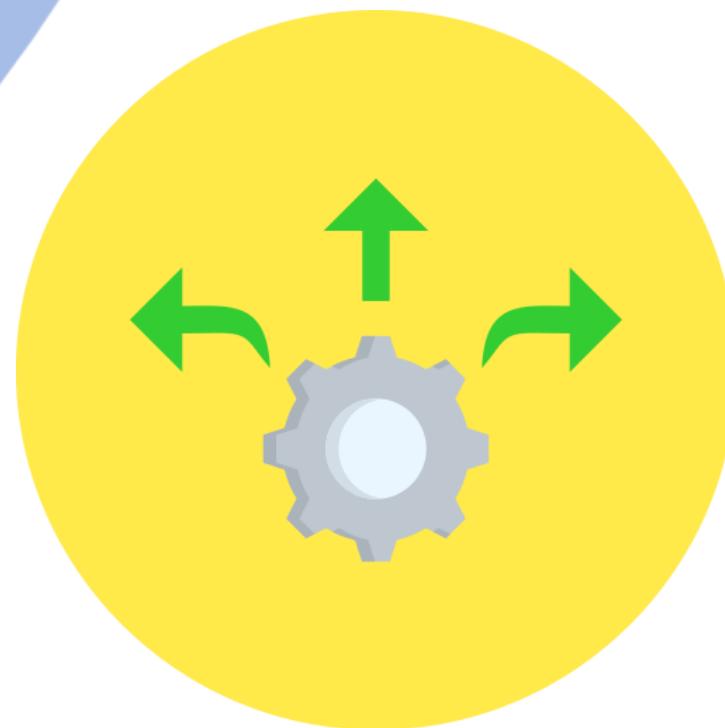
Cost Savings – Reduce expenses on IT staff and infrastructure upkeep



Introduction to Cloud

➤ Advantages of moving to the cloud

✓ Scalability & Flexibility



Instant Scaling – Expand or reduce resources in real time

No Hardware Delays – Avoid weeks/months of setup for on-premises infrastructure

Optimized Performance – Scale to meet demand without service disruptions

Ideal for Startups – Start small and grow as needed without upfront costs

Cost-Effective Expansion – Pay only for the resources used, adjusting as business needs change

Introduction to Cloud

➤ Advantages of moving to the cloud

✓ Global Accessibility & Remote Collaboration

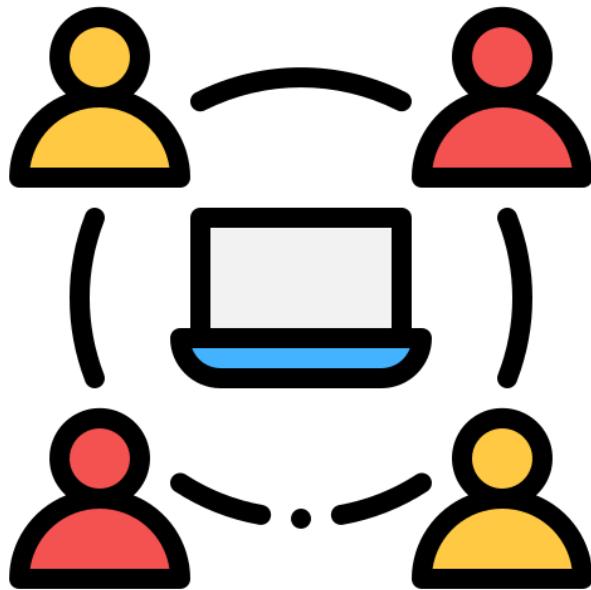
Access from Anywhere – Work from any location with an internet connection

No Geographic Limits – Collaborate across different time zones seamlessly

Supports Remote & Hybrid Work – Enables flexible work environments

Real-Time Collaboration – Teams can work on shared files and applications instantly

Increased Productivity – Employees stay connected and efficient, regardless of location



Introduction to Cloud

➤ Advantages of moving to the cloud

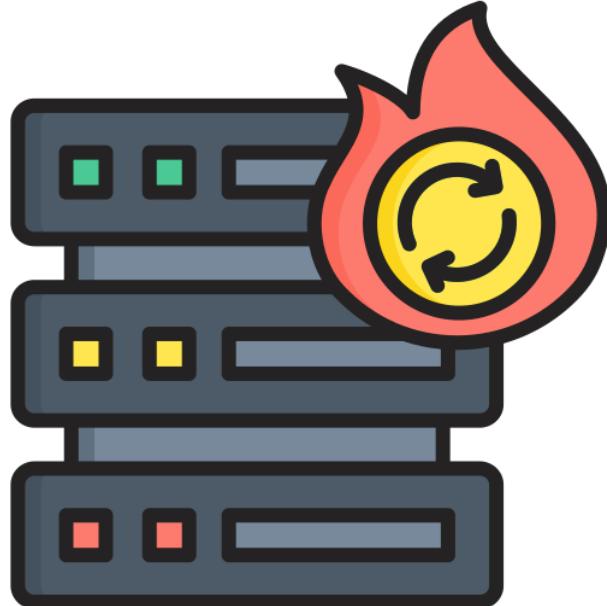
✓ Built-in Backup, Security & Disaster Recovery

Automated Backups – No manual effort required for data protection

Disaster Recovery – Minimize downtime with built-in failover solutions

Data Redundancy – Replication across multiple geographic locations

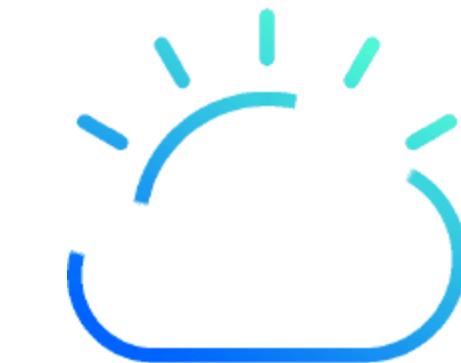
Enhanced Security – Encryption, access controls, and compliance measures



Introduction to Cloud

➤ Cloud Computing: From Product to Service

Cloud computing delivers computing as a service rather than a product



Introduction to AWS

Introduction to AWS

➤ What is AWS?

- ✓ Amazon Web Services
- ✓ Launched in 2006
- ✓ Provides on-demand technology services
- ✓ Pay-as-you-go pricing model
- ✓ No upfront hardware investments required
- ✓ Manages infrastructure for businesses
- ✓ Enables focus on growth and innovation
- ✓ Allows quick deployment and scaling of applications
- ✓ Helps in cost optimization

Introduction to AWS

➤ AWS: A Market Leader



Courtesy Gartner

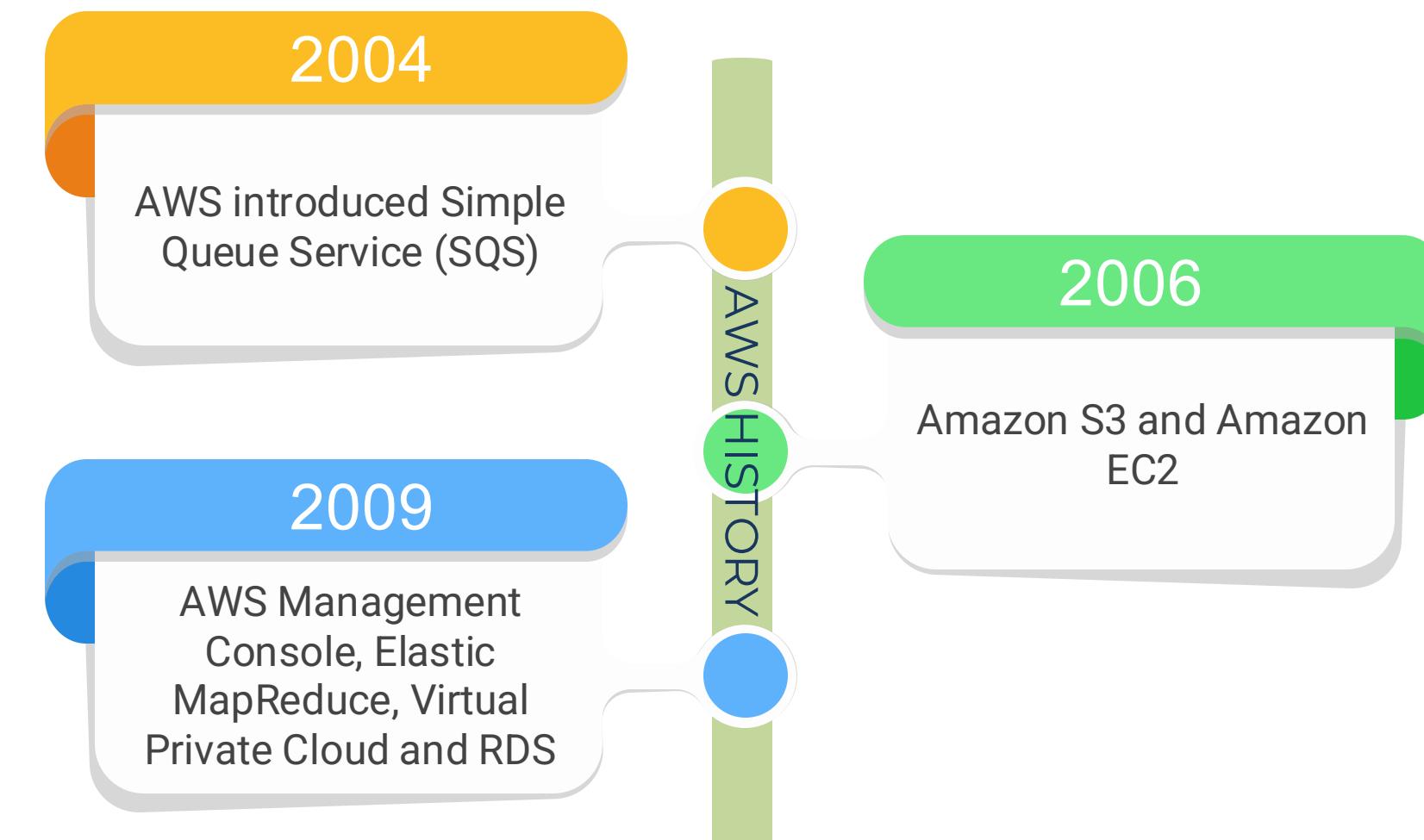
Introduction to AWS

➤ AWS History



Introduction to AWS

➤ AWS History



Introduction to AWS

➤ AWS History



Introduction to AWS

➤ AWS History



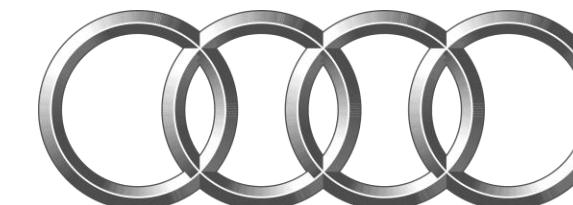
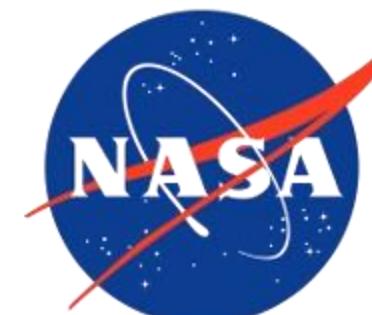
Introduction to AWS

➤ AWS History

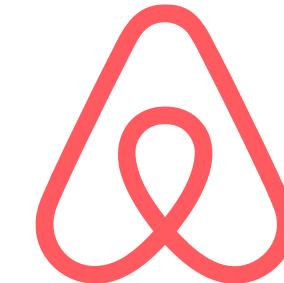


Introduction to AWS

➤ AWS History



Lenovo™



Coca-Cola



Introduction to AWS

➤ Key Benefits of AWS

✓ **Security**

- ✓ Strong data security with custom-built infrastructure
- ✓ 24/7 monitoring
- ✓ Encrypted data across the global network
- ✓ Full customer control over encryption, movement, and retention



Introduction to AWS

➤ Key Benefits of AWS

✓ Availability

- ✓ High availability with multiple Availability Zones
- ✓ Minimum of three independent Availability Zones per Region
- ✓ Isolates issues to prevent widespread impact
- ✓ Ensures continued operation even if one Region faces problems



Introduction to AWS

➤ Key Benefits of AWS

✓ Performance



- ✓ Low-latency, high-performance networking
- ✓ 400 GbE fiber backbone
- ✓ Local Zones for improved proximity to users
- ✓ Wavelength for ultra-low latency needs

Introduction to AWS

➤ Key Benefits of AWS

✓ Scalability

- ✓ Quickly scale resources as needed
- ✓ Reduces over-provisioning
- ✓ Minimizes costs
- ✓ Meets growing demands efficiently



Introduction to AWS

➤ Key Benefits of AWS

✓ **Flexibility**

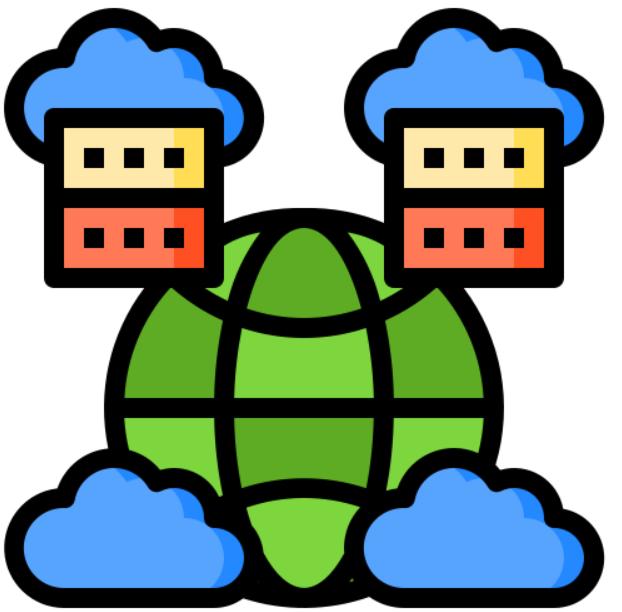
- ✓ Deploy workloads across Regions, Availability Zones, Local Zones, Wavelength, and Outposts
- ✓ Unified network and AWS services across all options



Introduction to AWS

➤ Key Benefits of AWS

✓ Global Reach



- ✓ Largest global infrastructure
- ✓ Deploy workloads closer to users
- ✓ Supports satellite services with AWS Ground Station

Regions, Availability Zones & Edge Locations

Regions, Availability Zones & Edge Locations

114 Availability Zones

36 launched Regions
each with multiple Availability Zones

700+ CloudFront POPs
and 13 Regional edge caches

Regions, Availability Zones & Edge Locations

➤ AWS Region

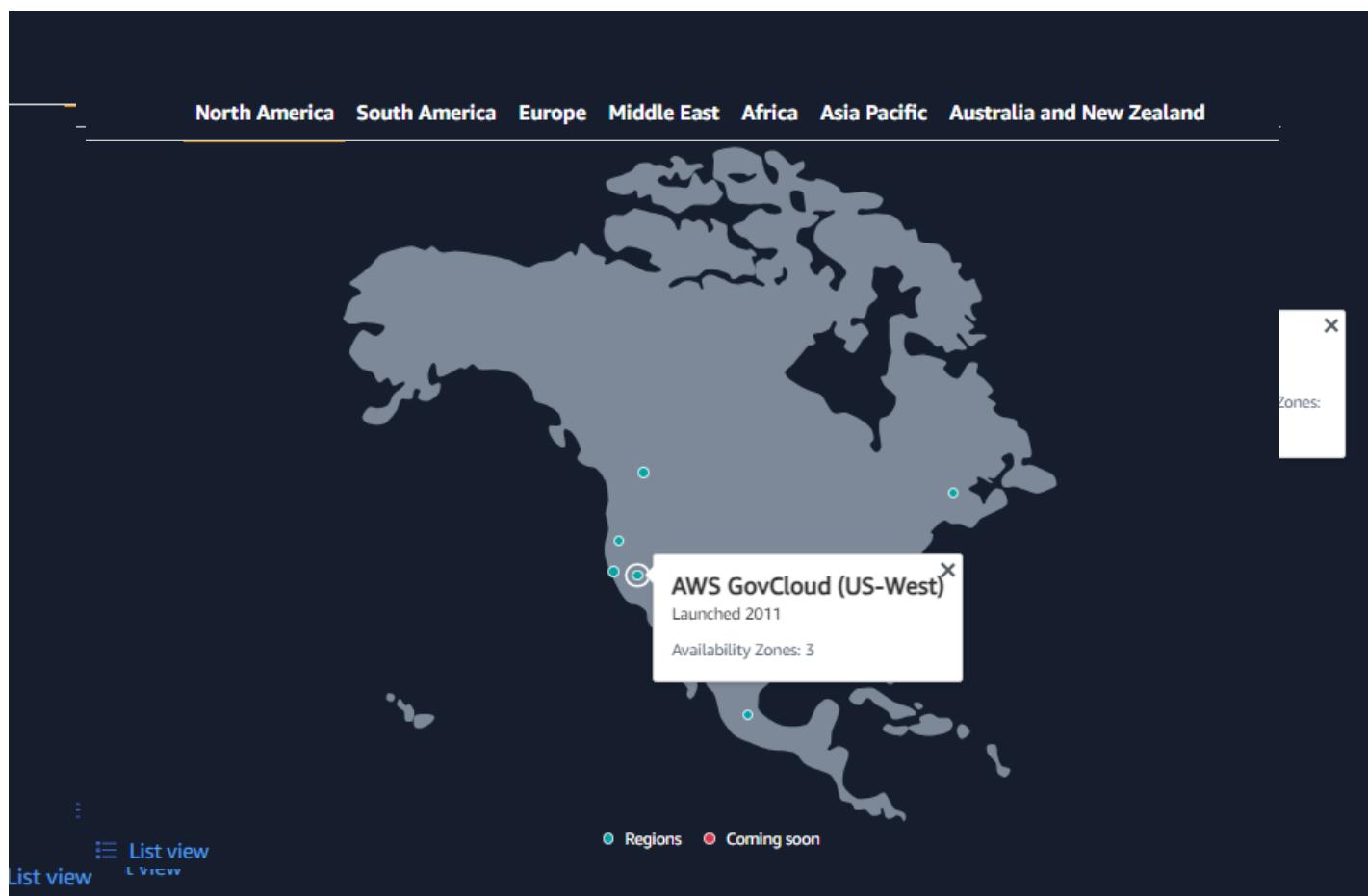
- ✓ Physical location with multiple data centers
- ✓ Each Region has at least three Availability Zones
- ✓ Availability Zones are isolated but connected

North America South America Europe Asia Pacific Africa Middle East

Regions, Availability Zones & Edge Locations

➤ AWS Region

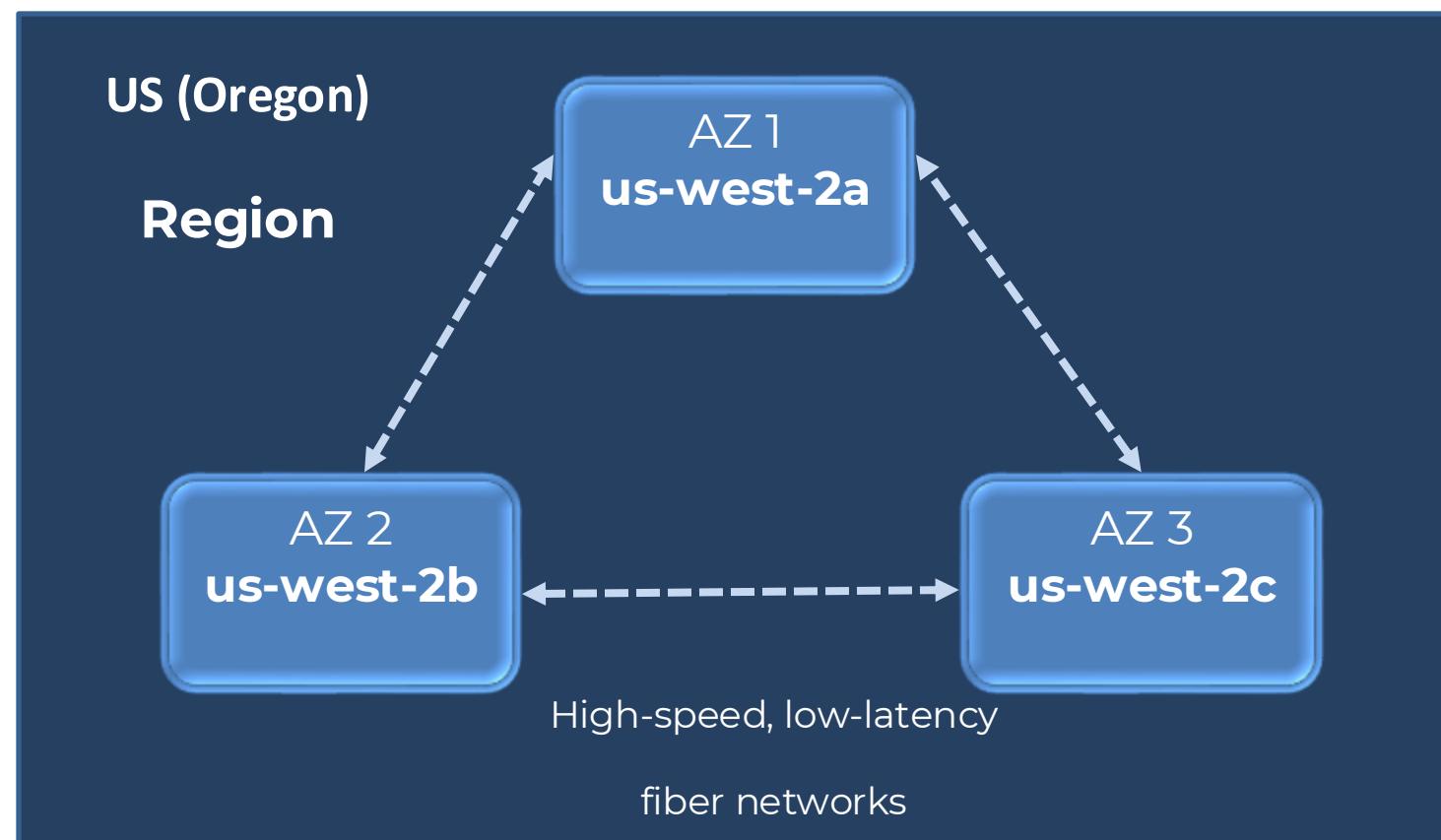
- ✓ Physical location with multiple data centers
- ✓ Each Region has at least three Availability Zones
- ✓ Availability Zones are isolated but connected



Regions, Availability Zones & Edge Locations

➤ Availability Zone (AZ)

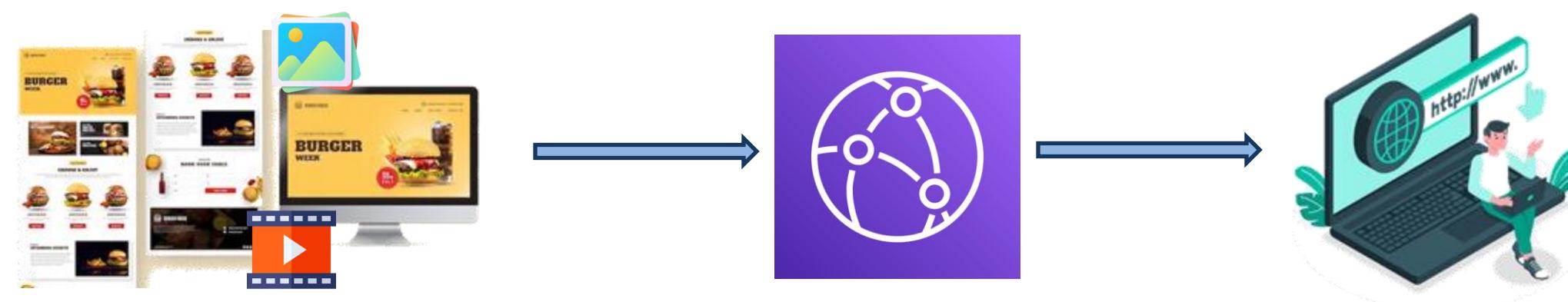
- ✓ One or more data centers within an AWS Region
- ✓ Each AZ has independent power, networking, and connectivity
- ✓ Ensures high availability and fault tolerance
- ✓ Connected with high-speed, low-latency fiber networks
- ✓ Provides scalability and resilience against local disruptions



Regions, Availability Zones & Edge Locations

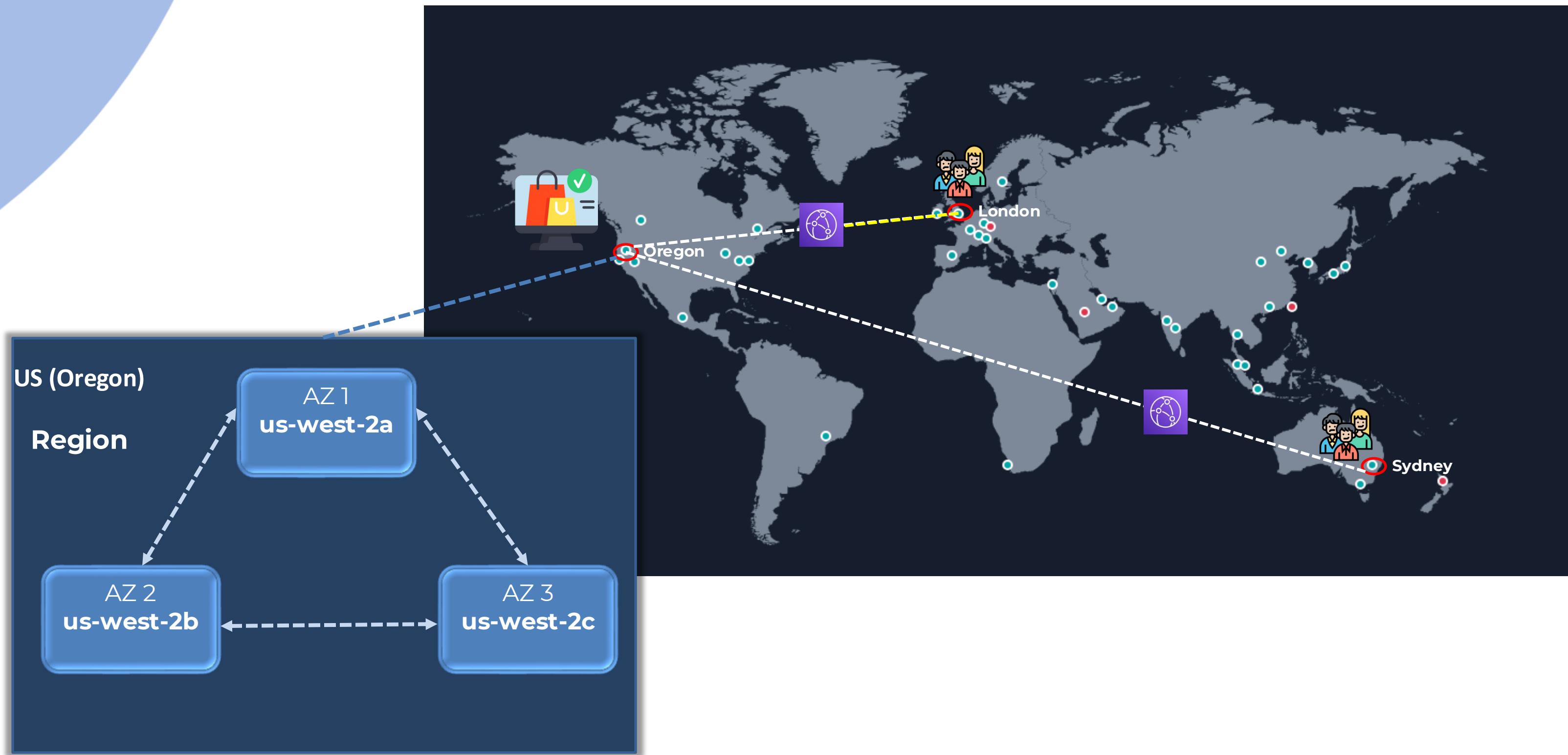
➤ Edge Location

- ✓ Part of AWS CloudFront network
- ✓ Delivers content quickly to users worldwide
- ✓ Caches images, videos, and web pages
- ✓ Brings content closer to users for faster access



Regions, Availability Zones & Edge Locations

➤ Real-World Example



Regions, Availability Zones & Edge Locations

➤ Evaluating Regions

- ✓ **Compliance** – Choose a Region that meets local regulations and data residency requirements
- ✓ **Latency** – Select a Region close to users to minimize network latency and improve performance
- ✓ **Cost** – AWS pricing varies by Region; opting for a lower-cost Region can reduce expenses
- ✓ **Services and Features** – New AWS services are deployed first in larger Regions, impacting availability

Regions, Availability Zones & Edge Locations

➤ In summary

- ✓ AWS Regions host data across multiple Availability Zones for reliability
- ✓ Edge Locations speed up content delivery to users
- ✓ Choosing a Region depends on compliance, latency, cost, and available services



Demonstration | AWS Account Setup & Dashboard Walkthrough

Introduction to EC2

Introduction to EC2

➤ Advantages of On-Premises Infrastructure

- ✓ Full control over hardware
- ✓ Predictable performance

Introduction to EC2

➤ Challenges of On-Premises Infrastructure

1. Physical Infrastructure

- ✓ Requires racks, cooling systems, and power management

2. High Upfront Investment & Deployment Delays

- ✓ Significant initial costs
- ✓ Deploying new servers can take weeks or even months

3. Security and Maintenance

- ✓ Demand dedicated IT teams

4. Scaling Limitations

- ✓ Requires purchasing and installing additional hardware
- ✓ Scaling isn't always efficient

Introduction to EC2

➤ Challenges of On-Premises Infrastructure

5. High Availability and Disaster Recovery

- ✓ Entirely falls on the organization
- ✓ Adds complexity and cost

6. Limited Agility

- ✓ Difficult to adapt quickly to changing business needs
- ✓ Launching new applications, expanding to new regions, or responding to traffic spikes requires lengthy procurement cycles and manual configurations
- ✓ Limits innovation and slows time-to-market

Introduction to EC2

➤ Amazon EC2: A Smarter Alternative

- ✓ Elastic Compute Cloud

□ Fast Deployment

- ✓ Launch virtual servers in minutes
- ✓ Eliminates costly hardware investments
- ✓ Removes long provisioning times

Introduction to EC2

➤ Amazon EC2: A Smarter Alternative

□ Flexible Instance Types

- ✓ Wide range of instance types
- ✓ Organizations can quickly adapt to different workloads
 - Compute-intensive tasks
 - Memory-heavy applications
 - General-purpose computing

Introduction to EC2

➤ Amazon EC2: A Smarter Alternative

□ Scalability with Auto Scaling

- ✓ Automatically adjusts the number of EC2 instances based on demand
- ✓ Keeps applications responsive during peak usage
- ✓ Reduces costs when traffic decreases
- ✓ Scale instantly and dynamically—no waiting weeks for new hardware

Introduction to EC2

➤ Amazon EC2: A Smarter Alternative

□ High Availability with ELB and Availability Zones

- ✓ Elastic Load Balancing (ELB) distributes traffic across multiple instances
- ✓ Prevents any single point of failure
- ✓ AWS operates across multiple availability zones
- ✓ Ensures seamless failover in case of infrastructure issues

Introduction to EC2

➤ Amazon EC2: A Smarter Alternative

□ Flexible and Resilient Storage Options

- ✓ Amazon EBS for persistent storage
- ✓ Amazon S3 for scalable object storage
- ✓ Amazon RDS for managed databases

Introduction to EC2

➤ Amazon EC2: A Smarter Alternative

□ Enhanced Security

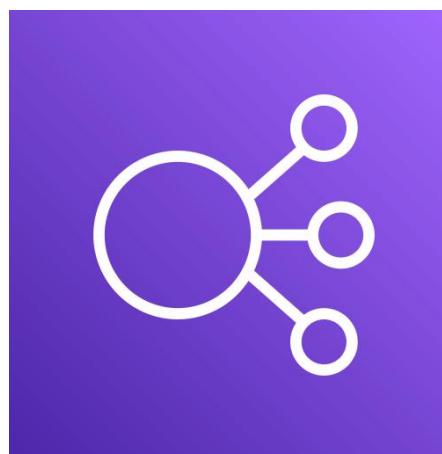
- ✓ IAM roles, security groups, and network ACLs
- ✓ Precise access control and network protection
- ✓ Minimizes risks associated with traditional data centers

Introduction to EC2

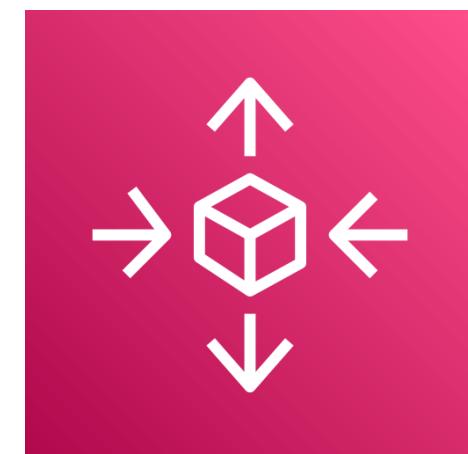
➤ Amazon EC2: A Smarter Alternative

□ Agility and Innovation

- ✓ Launch new products
- ✓ Respond to market shifts
- ✓ Expand globally
- ✓ Move faster, innovate sooner, and scale effortlessly



Load Balancing



Auto Scaling

Introduction to EC2

➤ What is EC2?

- ✓ **Amazon Elastic Compute Cloud** provides on-demand, scalable computing capacity



EC2

- ✓ Giving you the flexibility to choose and configure the operating system, software, and applications based on your needs

Introduction to EC2

➤ Features of EC2

- ✓ Amazon Machine Images
- ✓ Instance types
- ✓ Amazon EBS volumes
- ✓ Key pairs
- ✓ Security groups
- **Configure**
- **Secure**
- **Optimize**

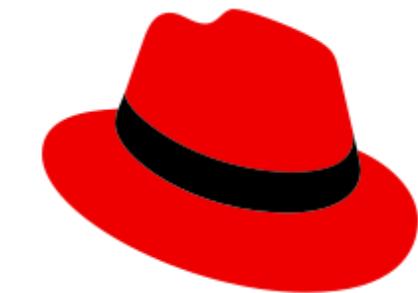
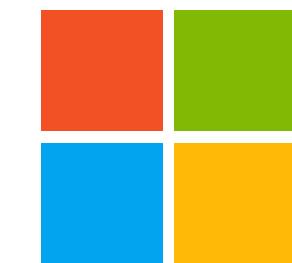
Introduction to EC2

➤ Amazon Machine Image

- ✓ Just like a physical server needs an OS, an EC2 instance requires a preconfigured template to run
- ✓ AMIs provide the OS, software, and settings to launch instances quickly and consistently

Introduction to EC2

- Amazon Machine Image
 - AWS offers a diverse range of AMIs



Introduction to EC2

➤ Amazon Machine Image

□ AWS offers a diverse range of AMIs

- ✓ **Amazon Linux** – Developed and optimized by **AWS** for performance and security
- ✓ **Ubuntu** – Maintained by **Canonical**, offering frequent updates and long-term support (LTS) versions
- ✓ **Windows Server** – Provided directly by **Microsoft**, with various editions tailored for enterprise applications
- ✓ **MacOS** – Available for Apple-based workloads, hosted on **dedicated Mac** instances
- ✓ **Red Hat Enterprise Linux (RHEL)** – Managed by **Red Hat**, designed for enterprise environments with full support and security updates
- ✓ **SUSE Linux Enterprise Server (SLES)** – Developed by **SUSE**, known for stability and optimized for cloud and enterprise workloads

Introduction to EC2

➤ Advantages of AMI

□ Custom AMIs for Efficiency

Why Use Custom AMIs?

- ✓ Pre-install applications, settings, and security configurations.
- ✓ Ensure quick and consistent deployment of multiple instances.
- ✓ Saves time and effort while maintaining uniform configurations.

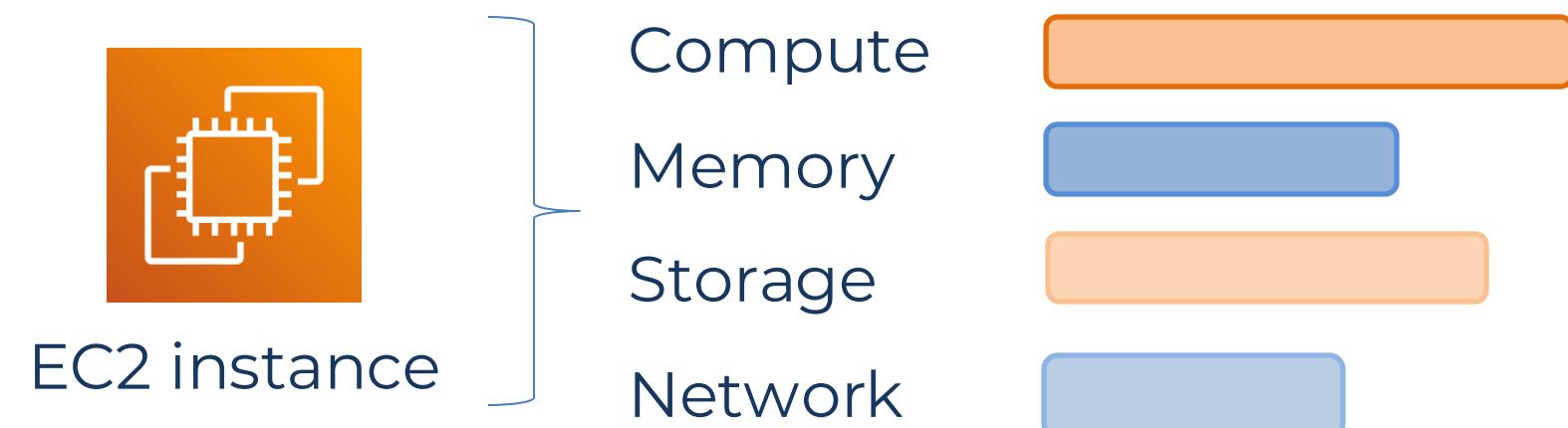
Introduction to EC2

➤ Amazon Machine Image

- Quick Start AMIs
- My AMIs
- AWS Marketplace AMIs
- Community AMIs

Introduction to EC2

➤ Instance Types



Introduction to EC2

➤ Instance Types

□ EC2 instances are divided into six primary categories

- ✓ **General Purpose** – Balanced compute, memory, and networking, ideal for web servers and code repositories
- ✓ **Compute Optimized** – High-performance processors for tasks like batch processing, media transcoding, and machine learning inference
- ✓ **Memory Optimized** – Designed for large in-memory workloads like real-time big data analytics
- ✓ **Storage Optimized** – Ideal for applications requiring high-speed access to large datasets, such as databases and big data processing
- ✓ **Accelerated Computing** – Uses GPUs or specialized hardware for AI, graphics processing, and complex calculations
- ✓ **High-Performance Computing (HPC)** – Optimized for running large-scale simulations and deep learning models

Introduction to EC2

➤ Instance Types

PAGE CONTENT

General Purpose

General Purpose

General purpose instances provide a balance of compute, memory and networking resources, and can be used for a variety of diverse workloads. These instances are ideal for applications that use these resources in equal proportions such as web servers and code repositories.

Compute Optimized

M8g	M7g	M7i	M7i-flex	M7a	Mac	M6g	M6i	M6in	M6a	M5	M5n	M5zn
M5a	M4	T4g	T3	T3a	T2							

Memory Optimized

[Amazon EC2 M8g instances](#) are powered by AWS Graviton4 processors. They deliver the best price performance in Amazon EC2 for general purpose workloads.

Accelerated Computing

Features:

- Powered by custom-built AWS Graviton4 processors
- Larger instance sizes with up to 3x more vCPUs and memory than M7g instances
- Features the latest DDR5-5600 memory
- Optimized for Amazon EBS by default
- Supports [Elastic Fabric Adapter \(EFA\)](#) on m8g.24xlarge, m8g.48xlarge, m8g.metal-24xl, and m8g.metal-48xl
- Powered by the [AWS Nitro System](#), a combination of dedicated hardware and lightweight hypervisor

Storage Optimized

HPC Optimized

Instance Features

Measuring Instance Performance

Instance size	vCPU	Memory (GiB)	Instance storage (GB)	Network bandwidth (Gbps)	Amazon EBS bandwidth (Gbps)
m8g.medium	1	4	EBS-only	Up to 12.5	Up to 10

You can visit: aws.amazon.com/ec2/instance-types

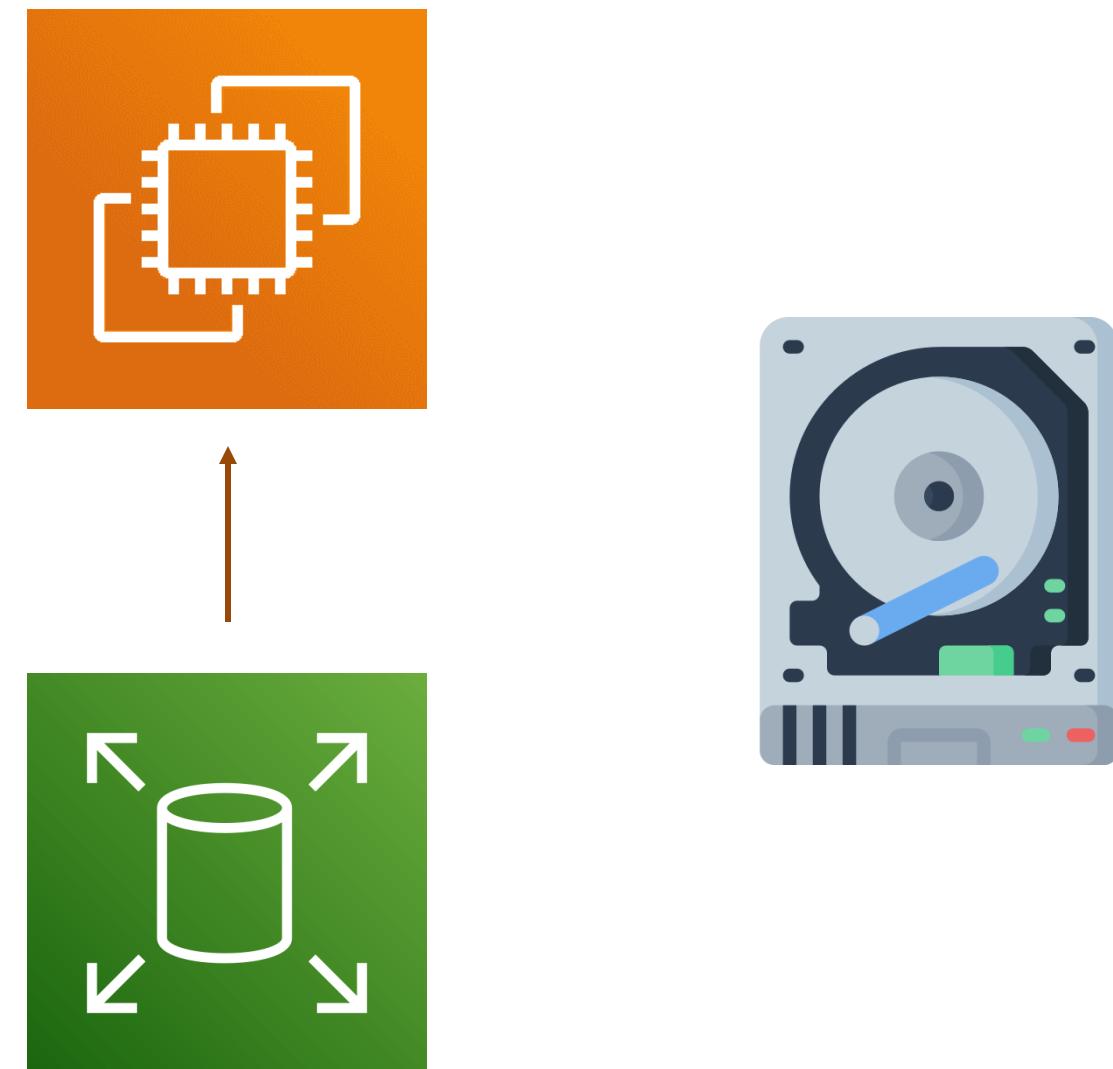
Introduction to EC2

➤ Amazon EBS volumes

- ✓ AWS automatically creates a **root volume** that contains the **boot image**
- ✓ One root volume
- ✓ Add more storage anytime
- ✓ Even after launch
- ✓ Without stopping the instance

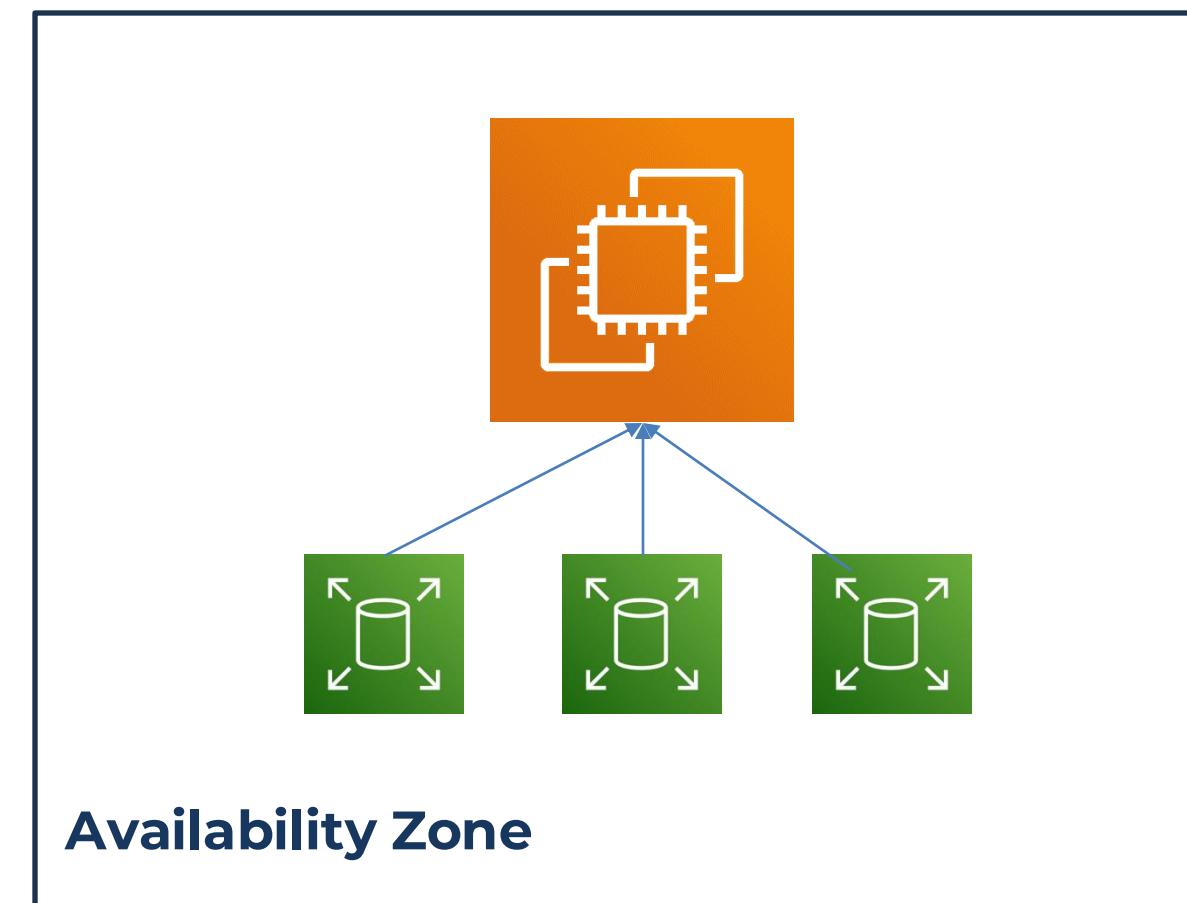
Introduction to EC2

➤ Amazon EBS volumes



Introduction to EC2

➤ Amazon EBS volumes



Some volume and instance types even support Multi-Attach

Introduction to EC2

➤ Amazon EBS volumes

- **AWS recommends using EBS-backed AMIs**
 - ✓ Faster launch times and persistent storage
 - ✓ EBS volumes are flexible
 - You can resize or increase performance
 - You can create EBS snapshots to back up data or move it

Accounts

Regions

Availability
Zones

Introduction to EC2

➤ Amazon EBS volumes

- ✓ By default, EBS root volumes are deleted when the instance is terminated

DeleteOnTermination = False

Introduction to EC2

➤ EBS Volume Types

- **SSD-backed volumes**
- **HDD-backed volumes**

Introduction to EC2

➤ EBS Volume Types

□ SSD-backed volumes

- General Purpose SSD (**gp3, gp2**)
- Provisioned IOPS SSD (**io1, io2**)

□ HDD-backed volumes

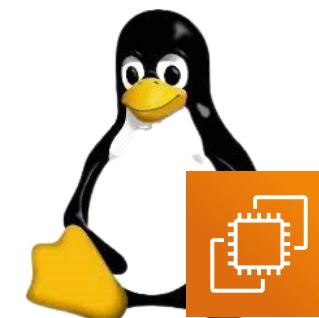
- Throughput Optimized HDD (**st1**)
- Cold HDD (**sc1**)

Only gp2/gp3 and io1/io2 can be used as boot volumes

Introduction to EC2

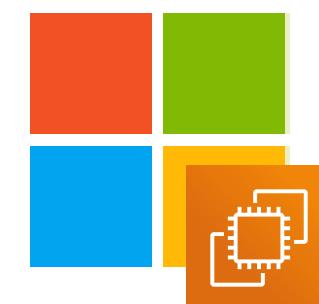
➤ Key Pairs

- ✓ **Public Key** – Stored on your instance by AWS
- ✓ **Private Key** – Stored securely on your computer and used for authentication



- ✓ Private key allows you to securely SSH

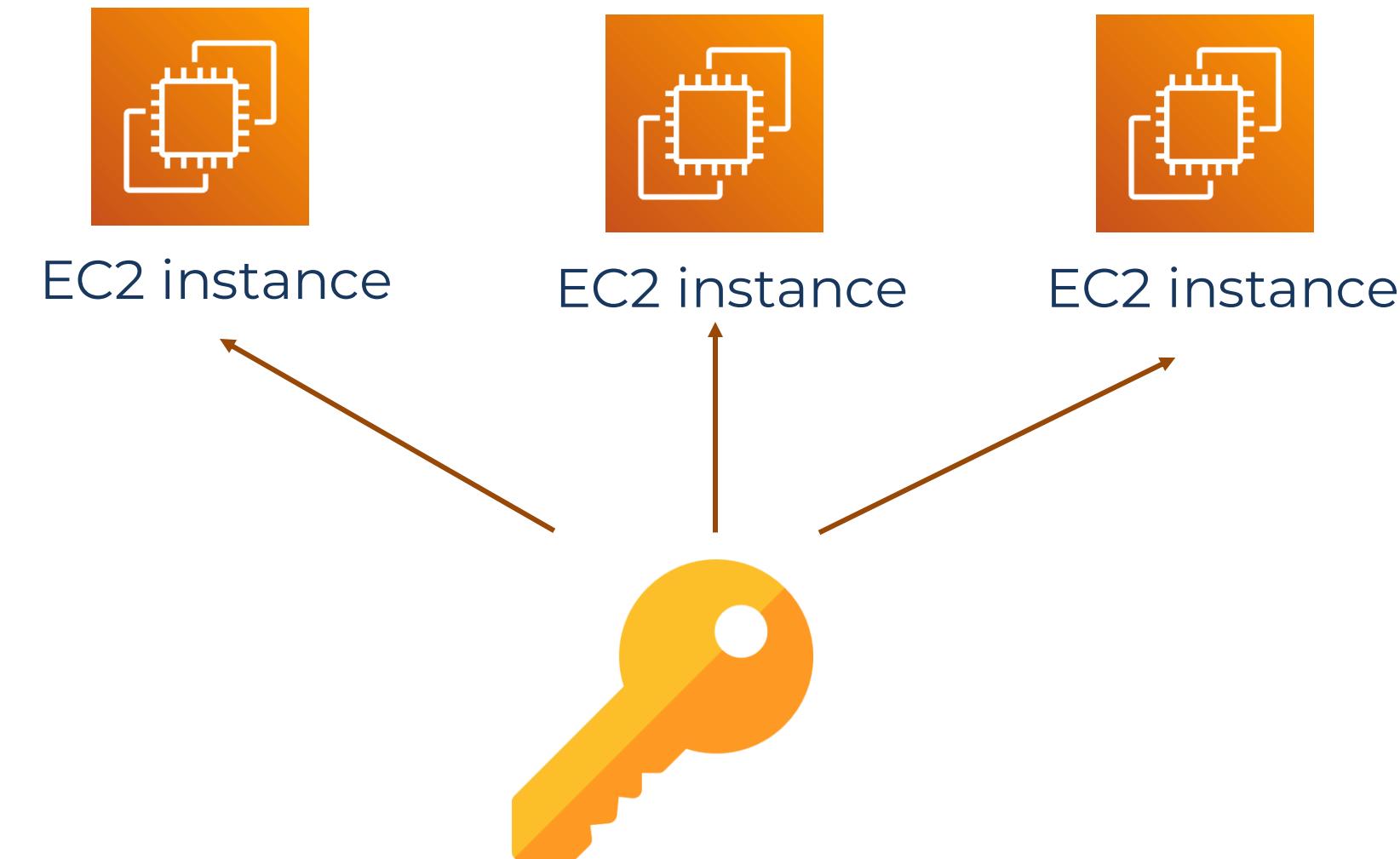
`~/.ssh/authorized_keys`



- ✓ Private key is required to decrypt the administrator password

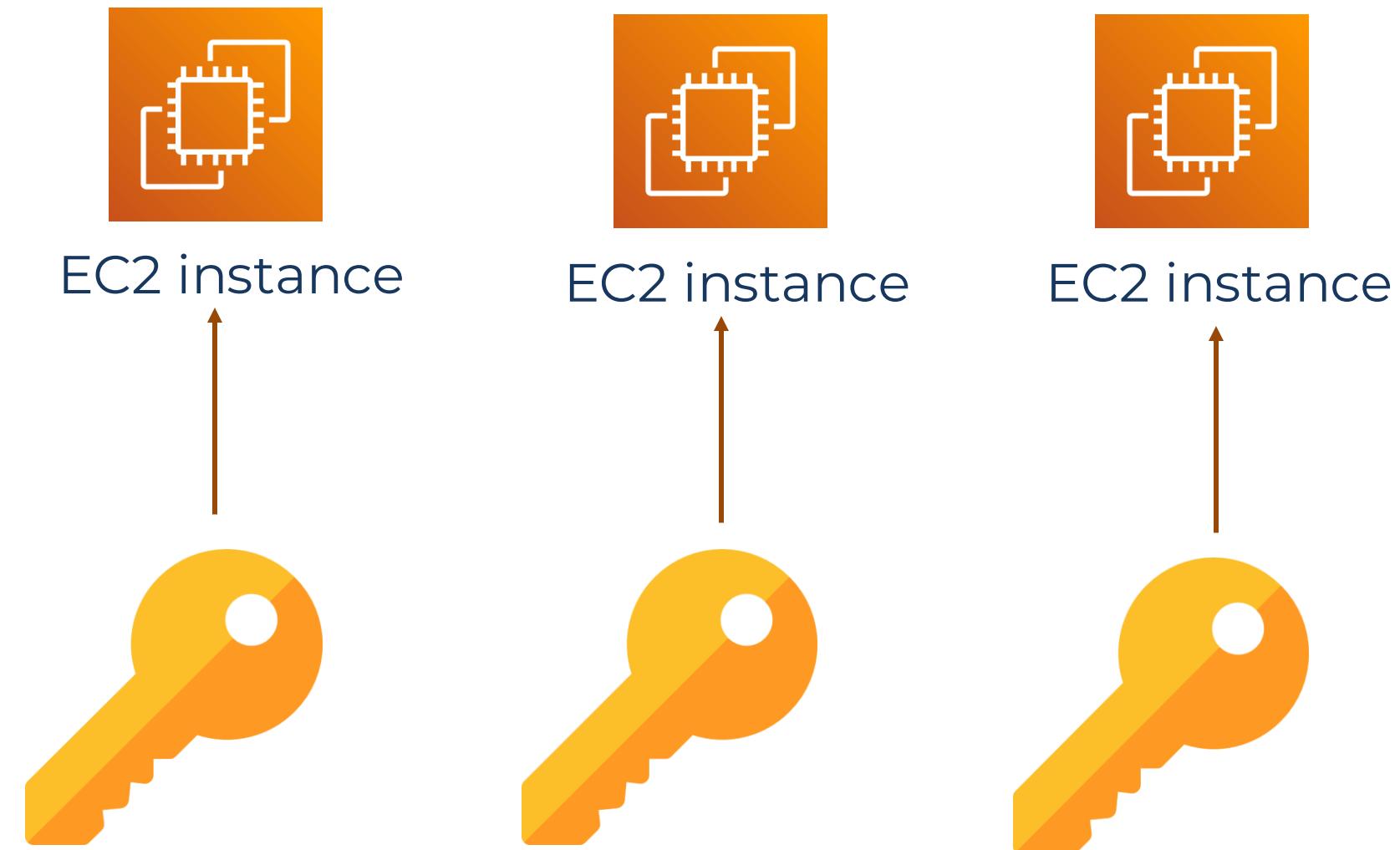
Introduction to EC2

➤ Key Pairs



Introduction to EC2

➤ Key Pairs



Introduction to EC2

➤ Key Pairs

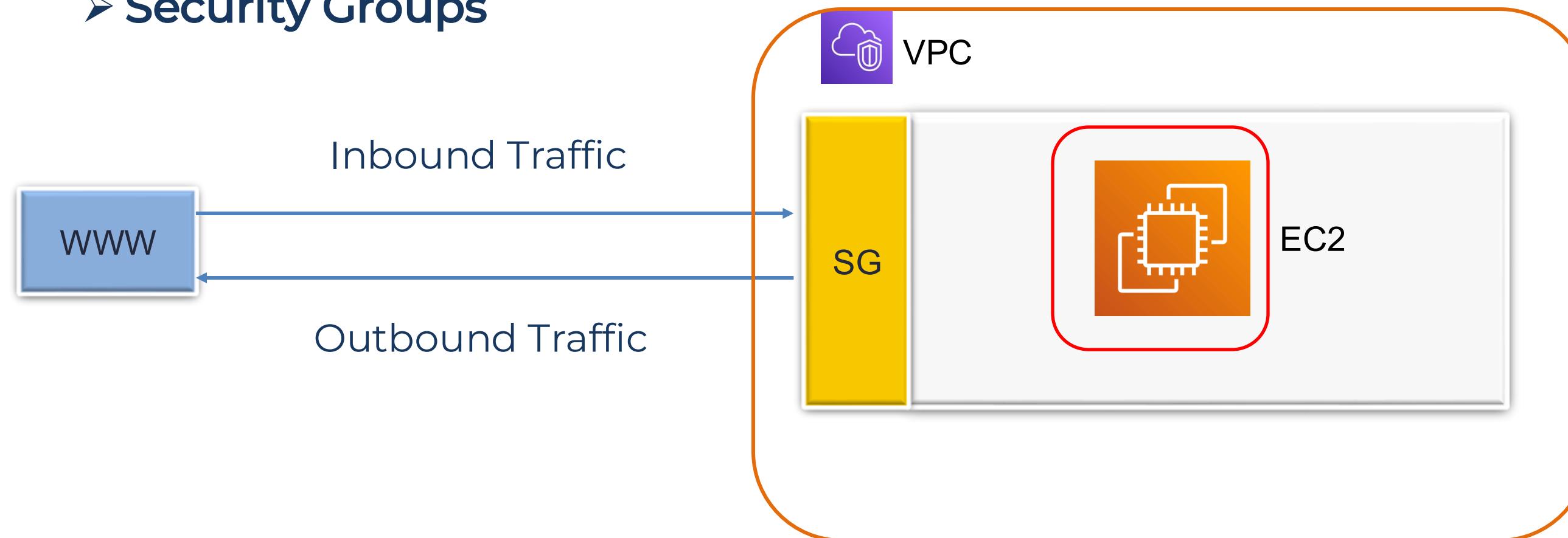


Private Key

- ✓ Amazon EC2 does not keep a copy
- ✓ Store it securely

Introduction to EC2

➤ Security Groups



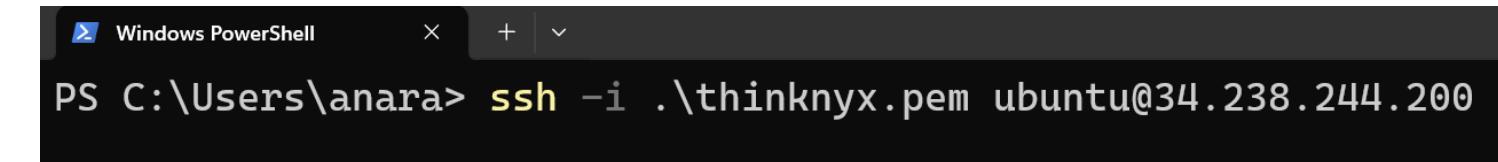
- ✓ Security group acts as a virtual firewall for EC2 instances
- ✓ Controls the inbound and outbound traffic of EC2 instances
- ✓ Security groups operate only within the VPC
- ✓ You can modify security group rules anytime, and changes apply immediately

Introduction to EC2

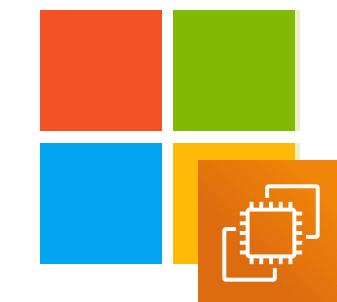
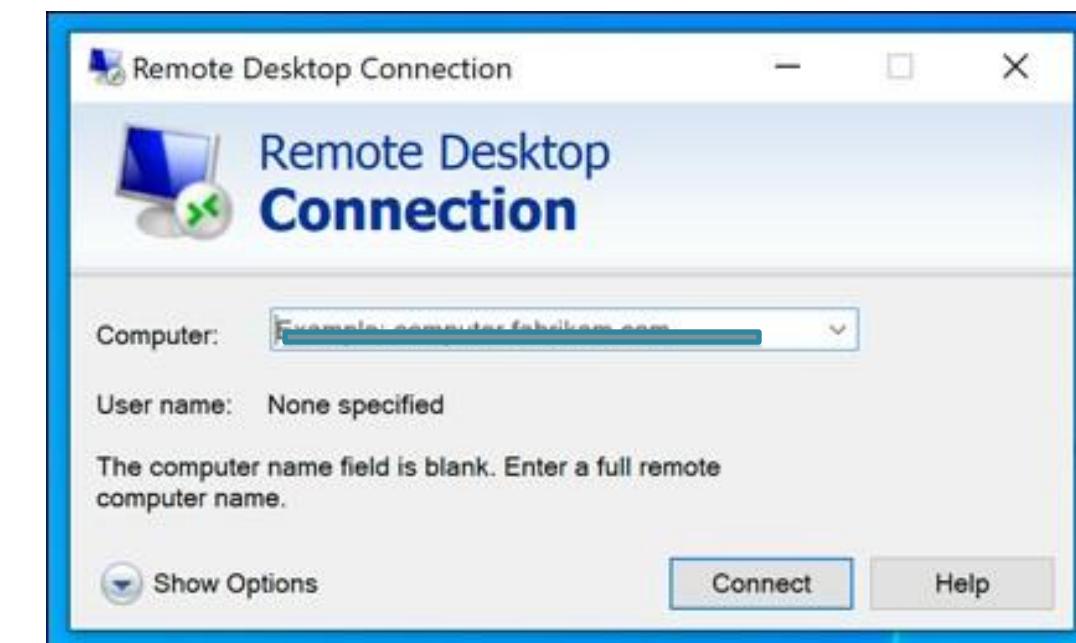
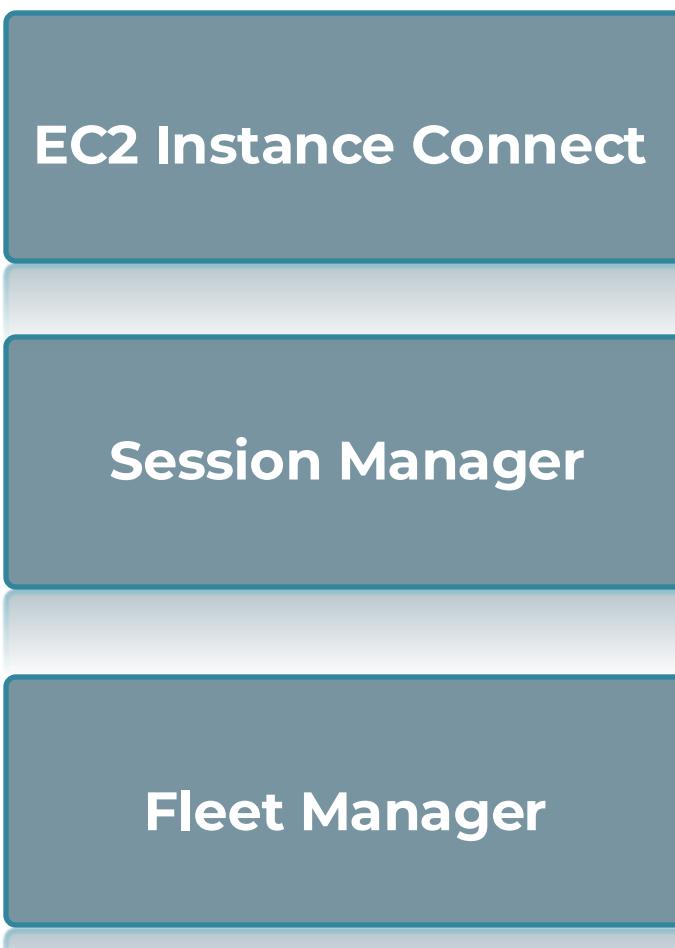
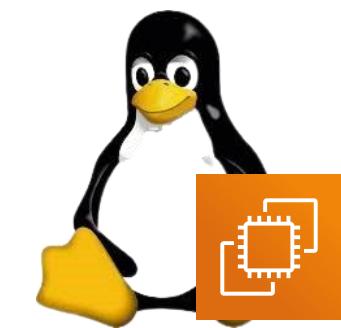
➤ Connect to EC2 instance



Amazon EC2

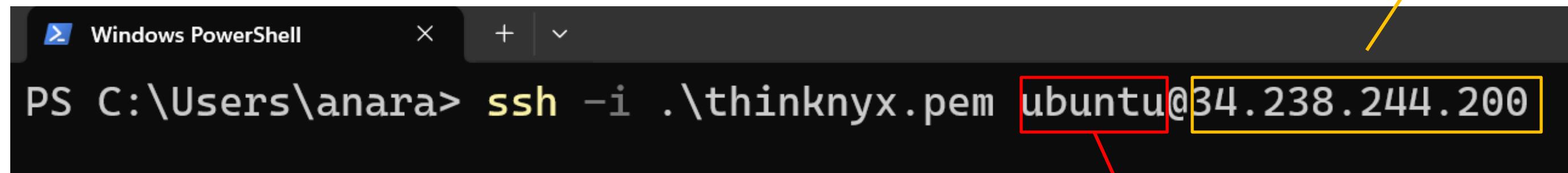


```
PS C:\Users\anara> ssh -i .\thinknyx.pem ubuntu@34.238.244.200
```

A screenshot of a Windows PowerShell window. The command entered is "ssh -i .\thinknyx.pem ubuntu@34.238.244.200".

Introduction to EC2

➤ Connect to EC2 instance



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The command entered is "PS C:\Users\anara> ssh -i .\thinknyx.pem ubuntu@34.238.244.200". A red box highlights the word "ubuntu" in the command, and a yellow box highlights the IP address "34.238.244.200". An arrow points from the text "Public IP" to the yellow box, and another arrow points from the text "Username" to the red box.

- For an Amazon Linux AMI, the username is `ec2-user`.
- For a CentOS AMI, the username is `centos` or `ec2-user`.
- For a Debian AMI, the username is `admin`.
- For a Fedora AMI, the username is `fedora` or `ec2-user`.
- For a RHEL AMI, the username is `ec2-user` or `root`.
- For a SUSE AMI, the username is `ec2-user` or `root`.
- For an Ubuntu AMI, the username is `ubuntu`.
- For an Oracle AMI, the username is `ec2-user`.
- For a Bitnami AMI, the username is `bitnami`.

Introduction to EC2

➤ Understanding Public vs. Private vs. Elastic IPs

Private IP

- ✓ Internal VPC communication
- ✓ Static (doesn't change when instance stops/starts)

Public IP

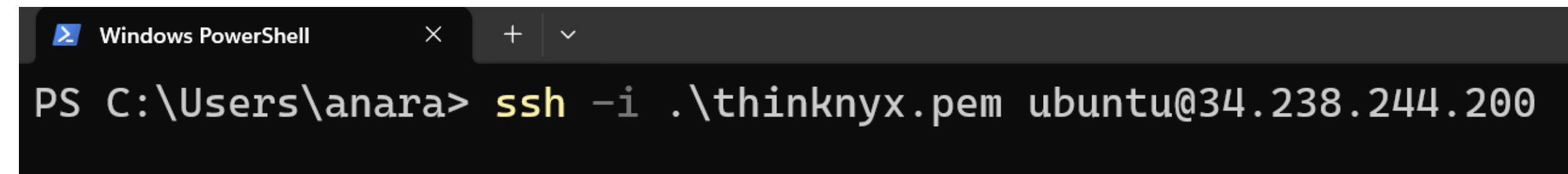
- ✓ Internet access
- ✓ Dynamic (changes when instance stops/starts)

Elastic IP (EIP)

- ✓ Static public IP
- ✓ Persistent (remains the same until released)

Introduction to EC2

Upcoming Demonstration



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The command entered is "PS C:\Users\anara> ssh -i .\thinknyx.pem ubuntu@34.238.244.200". The window has a dark theme with white text.

```
PS C:\Users\anara> ssh -i .\thinknyx.pem ubuntu@34.238.244.200
```



Demonstration | Launch an EC2 instance

Summary

- Infrastructure setup completed
- Covered cloud basics and importance
- Explored AWS fundamentals: Regions & Availability Zones
- Set up AWS account
- Launched EC2 instance
- Different EC2 names may appear in demos
- Variations due to regions, accounts, security groups, key pairs
- Core concepts and process remain the same



Demonstration | Updating Cloud Progress in GitHub Projects



Build and Scale with AWS Cloud – A Hands-on Beginners Guide

THANK YOU



Thinknyx Technologies

Linux for DevOps

Section - 6





Linux | Introduction to Linux Section



Introduction to Linux Section

Basic commands

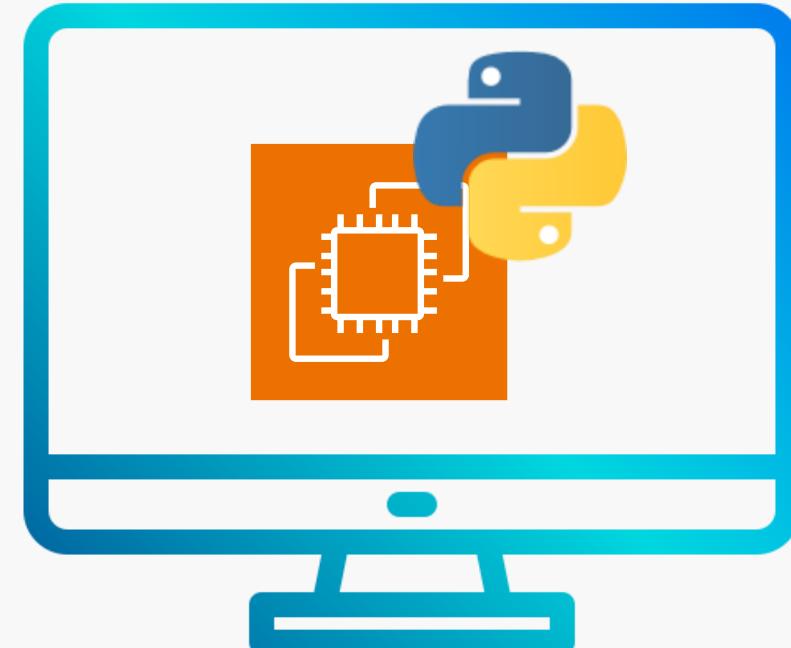
Working with utilities

Navigating directories and files

Viewing file contents

Using text editors

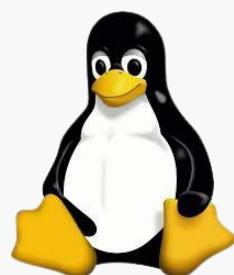
- File ownership
- Package management
- Process management
- Managing services



Not

e:

Most of the videos here, except for the final deployment demo, are optional. They are designed with beginners in mind. If you're new to Linux, we recommend going through them, as they'll build your awareness of commands, files, and system behavior—skills that will help you throughout your DevOps journey

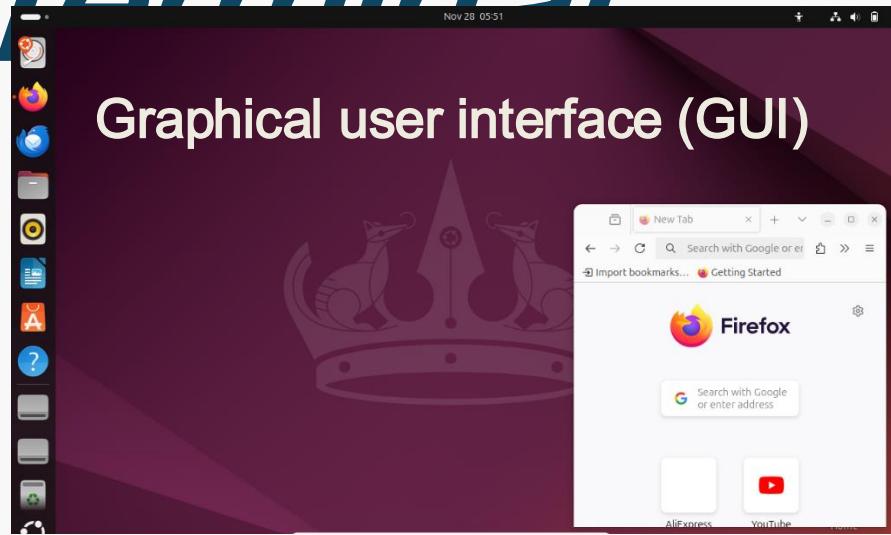




Linux | Getting Started with Terminal



Getting Started with Terminal



- ✓ Visual and Intuitive Navigation
- ✓ Beginner-Friendly
- ✓ Point-and-Click Interface

- ✓ Powerful Text-Based Approach
- ✓ Speed and Precision
- ✓ Greater Control
- ✓ Preferred by advanced users and system administrators



Getting Started with Terminal

➤ Basic Elements of a Shell Command

- ✓ The shell is a program that interprets user commands
- ✓ Commands are names of programs or scripts installed on the system
- ✓ Each command may include options and arguments



Getting Started with Terminal

➤ Basic Elements of a Shell Command

Syntax:

Command [Options] [Arguments]

Command	Description
ls	Lists files and directories in the current directory
ls -l	Lists files and directories in a detailed (long) format in the current directory
ls -l /dev	Lists files and directories in the specified directory (/dev) with detailed information

```
ubuntu@thinknyxlinux:~/dev$ ls -l /dev
total 0
crw-r--r-- 1 root root      10, 235 Nov 14 12:06 autofs          vcsu3
drwxr-xr-x 2 root root      300 Nov 14 12:07 block           vcsu4
crw----- 1 root root      10, 234 Nov 14 12:06 btrfs-control   vcsu5
drwxr-xr-x 2 root root     2620 Nov 14 12:07 char            vcsu6
crw--w---- 1 root tty       5,  1 Nov 14 12:07 console         vfio
lrwxrwxrwx 1 root root      11 Nov 14 12:06 core -> /proc/kcore  vga_arbiter
drwxr-xr-x 3 root root      60 Nov 14 12:06 cpu             vhost-net
crw----- 1 root root     10, 122 Nov 14 12:06 cpu_dma_latency xen
crw----- 1 root root     10, 203 Nov 14 12:06 cuse            xvda
drwxr-xr-x 6 root root     120 Nov 14 12:07 disk           xvda1
drwxr-xr-x 2 root root      60 Nov 14 12:06 dma_heap        xvda14
crw----- 1 root root     10, 125 Nov 14 12:06 encryptfs      xvda15
lrwxrwxrwx 1 root root      13 Nov 14 12:06 fd -> /proc/self/fd xvda16
crw-rw-rw- 1 root root      1,  7 Nov 14 12:06 full           zero
crw-rw-rw- 1 root root     10, 229 Nov 14 12:06 fuse          zfs
crw----- 1 root root     10, 228 Nov 14 12:06 hpet
drwxr-xr-x 2 root root      0 Nov 14 12:06 hugepages
```



Getting Started with Terminal

➤ Basic Commands

Command	Description	Syntax	Usage
echo	Prints the string(s) to standard output	echo <string(s)>	echo "Hello, Linux!"
uname	Prints the Systems Information	uname [options]	uname -a
whoami	Prints the username associated with the current effective user ID	whoami	whoami
who	Prints information about users who are currently logged in	who	who
uptime	Tells how long the system has been running	uptime	uptime
date	Prints or sets the system date and time	date	date





Linux | Working with Terminal Utilities



Working with Terminal Utilities

➤ Basic Commands

Command	Description	Syntax	Usage
clear	Clears the terminal screen	clear	clear
history	Lists previously executed commands	history	history
man	System's manual pager	man <command>	man clear
whatis	Displays one-line manual page descriptions	whatis <command>	whatis clear
whereis	Locates the binary, source, and manual page files for a command	whereis <command>	whereis clear
which	Locates the path of binary, source file for a command	which <command>	which ls





Linux | Working with Directories



Working with Directories

➤ Basic Commands

Command	Description	Syntax	Usage
pwd	Prints the full filename of the current working directory	pwd	pwd
cd	Changes directory	cd <directory>	cd /tmp
ls	List information about the FILES of the current directory by default	ls	ls
mkdir	Makes/Creates directory	mkdir <directory>	mkdir thinknyx
rmdir	Removes empty directories	rmdir <directory>	rmdir thinknyx

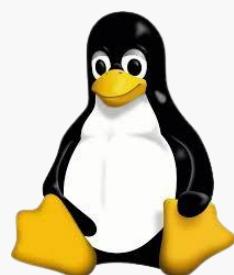


Working with Directories

➤ Understanding Absolute and Relative Paths

Absolute and Relative Paths:

- ✓ Two ways to locate files and directories
- ✓ Simplify system navigation



Working with Directories

➤ Understanding Absolute and Relative Paths

Absolute Paths:

- ✓ Starts from the root directory (/)
- ✓ Provides the full path to a file or folder
- ✓ Exact location, independent of the current directory
- ✓ Use cd to navigate



```
cd /usr/bin
```



Working with Directories

➤ Understanding Absolute and Relative Paths

Relative Paths:

- ✓ Starts from the current directory
- ✓ Doesn't begin with (/)

/home/user/documents Current directory

/home/user Parent directory

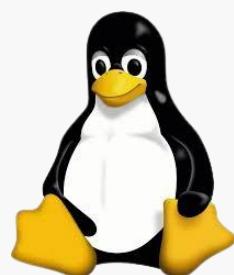
cd ..

.	Current directory
..	Parent directory
~	Home directory

Deeper directory /home/user/documents/work

/home/user

cd ../../..



Working with Directories

➤ Understanding Absolute and Relative Paths

When to Use Which?

- **Relative Path:** Quick navigation between nearby directories

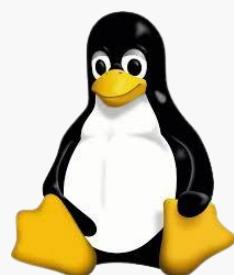
```
/home/user/documents to /home/user/projects
```

```
cd ../projects
```

- **Absolute Path:** Reliable for deeper or fixed locations

```
/usr/bin
```

```
cd /usr/bin
```



Working with Directories

Character	Type
d	Directory
-	Regular file
l	Symbolic links
b	Block devices
c	Character devices
p	Named pipes(FIFOs)
s	Sockets





Linux | Working with Files



Working with Files

➤ Basic Commands

Command	Description	Syntax	Usage
touch	Creates an empty file;	touch <filename>	touch thinknyx.txt
file	Determines file type	file <filename>	file thinknyx.txt
cp	Copies files and directories	cp <src_filename> <dest_filename>	cp thinknyx.txt test.txt
mv	Moves or renames files	mv <src_filename> <dest_filename>	mv test.txt result.txt
rm	Removes files or directories	rm <filename>	rm result.txt





Linux | Working with File Contents



Working with File Contents

➤ Basic Commands

Command	Description	Syntax	Usage
cat	Concatenates files and prints on the standard output	cat <file1> <file2>	cat file1.txt file2.txt
tac	Concatenates files and prints the contents on the standard output in reverse order	tac <file1> <file2>	tac file1.txt file2.txt
head	Prints the first 10 lines of the file to standard output by default	head <filename>	head file1.txt
tail	Prints the last 10 lines of the file to standard output by default	tail <filename>	tail file1.txt
more	Prints file content page-by-page	more <filename>	more file1.txt
less	Prints file content page-by-page in reverse	less <filename>	less file1.txt





Linux | Text Editors

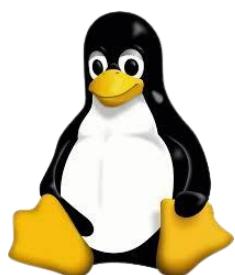


Text

Editors

- ✓ Text editors are crucial for editing files directly in the terminal
- ✓ Two popular text editors: Nano and Vi

- **Nano:** Beginner-friendly and straightforward
- **Vi:** Advanced features for power users



Text Editors

➤ Nano Editor

- ✓ Lightweight and easy-to-use text editor
- ✓ Ideal for quick edits and commonly used by beginners
- ✓ Open a file by using: `nano filename.txt`

Action	Shortcut
Save a file	CTRL + O
Exit Nano	CTRL + X
Search for text	CTRL + W
Cut a line	CTRL + K
Paste a line	CTRL + U



Text Editors

➤ Vi Editor

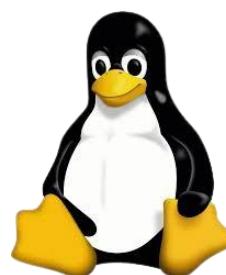
- ✓ Widely used by advanced Linux users for efficient text editing
- ✓ Vi stands for Visual Editor
- ✓ Vim (Vi Improved) is the enhanced version of vi, adding more features, flexibility, and functionality
- ✓ Ideal for editing, navigating, and manipulating large amounts of text

Mode	Description
Normal Mode Or Command mode	Default mode for navigation and commands
Insert Mode	For adding or editing text
Ex-Command Mode or Last line mode	For executing operations like saving, quitting & searching



Text Editors

/pattern	Searches forward for the pattern from cursor position
?pattern	Searches backward for pattern from cursor position
:%s/pattern/replacement/g	Replace all occurrences of 'pattern' with 'replacement' in the file
:%s/pattern/replacement/gc	Replace all occurrences of 'pattern' with confirmation before each change

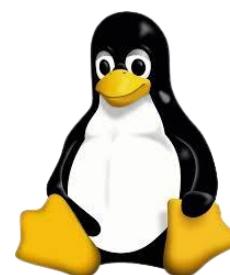


Text Editors

➤ Advanced Vim Features

Action	Command
Delete a line	dd
Delete 3 lines	3dd
Delete character at cursor	x
Undo changes	u
Redo changes	CTRL + R

Action	Command
Copy	yy
Paste below cursor	p
Paste above cursor	P
Append to end of line	Shift + A



Text Editors

Command	Functionality	Timestamp Behavior
:wq	Saves changes and exits the editor	Always updates the timestamp to current time
:x	Saves changes and exits, but only updates the timestamp if changes were made	Updates timestamp only if changes were made





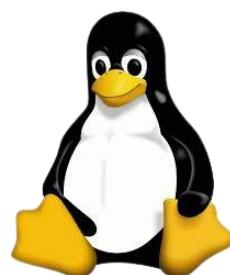
Linux | File Ownership



File Ownership

```
ubuntu@ip-172-31-37-18:/tmp$ ls -l
total 24
drwx----- 2 root root 4096 Nov 19 08:36 snap-private-tmp
drwx----- 3 root root 4096 Nov 19 08:36 systemd-private-86398170d5c448128a49d3f65fbddcfe-ModemManager.service-QvKYeo
drwx----- 3 root root 4096 Nov 19 08:36 systemd-private-86398170d5c448128a49d3f65fbddcfe-chrony.service-QZNxxE
drwx----- 3 root root 4096 Nov 19 08:36 systemd-private-86398170d5c448128a49d3f65fbddcfe-polkit.service-UAnXSe
drwx----- 3 root root 4096 Nov 19 08:36 systemd-private-86398170d5c448128a49d3f65fbddcfe-systemd-logind.service-jjDUTR
drwx----- 3 root root 4096 Nov 19 08:36 systemd-private-86398170d5c448128a49d3f65fbddcfe-systemd-resolved.service-TYd6lT
ubuntu@ip-172-31-37-18:/tmp$
```

Link count Group
File type
Permissions Owner File size Last modification time File name



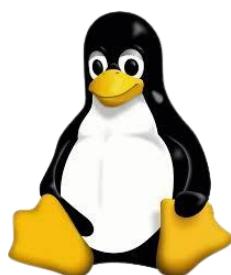
File Ownership

```
ubuntu@ip-172-31-37-18:/tmp$ ls -l
total 24
drwx----- 2 root root 4096 Nov 19 08:36 snap-private-tmp
drwx----- 3 root root 4096 Nov 19 08:36 systemd-private-86398170d5c448128a49d3f65fbddcfec-ModemManager.service-QvKYeo
drwx----- 3 root root 4096 Nov 19 08:36 systemd-private-86398170d5c448128a49d3f65fbddcfec-chrony.service-QZNxxE
drwx----- 3 root root 4096 Nov 19 08:36 systemd-private-86398170d5c448128a49d3f65fbddcfec-polkit.service-UAnXSe
drwx----- 3 root root 4096 Nov 19 08:36 systemd-private-86398170d5c448128a49d3f65fbddcfec-systemd-logind.service-jjDUTR
drwx----- 3 root root 4096 Nov 19 08:36 systemd-private-86398170d5c448128a49d3f65fbddcfec-systemd-resolved.service-TYd6lT
ubuntu@ip-172-31-37-18:/tmp$
```

File size

Last modification time

File name



File Ownership

```
ubuntu@thinknyxlinux:/tmp$ ls -lh
total 24K
drwx----- 2 root root 4.0K Nov 22 08:44 snap-private-tmp
drwx----- 3 root root 4.0K Nov 22 08:45 systemd-private-311f0da5651842b38945cabffffd9098-ModemManager.service-qXDUSo
drwx----- 3 root root 4.0K Nov 22 08:45 systemd-private-311f0da5651842b38945cabffffd9098-chrony.service-9qEvUa
drwx----- 3 root root 4.0K Nov 22 08:45 systemd-private-311f0da5651842b38945cabffffd9098-polkit.service-TZQXIv
drwx----- 3 root root 4.0K Nov 22 08:45 systemd-private-311f0da5651842b38945cabffffd9098-systemd-logind.service-Ep9JBB
drwx----- 3 root root 4.0K Nov 22 08:44 systemd-private-311f0da5651842b38945cabffffd9098-systemd-resolved.service-E4Gr0u
```

File size



File Ownership

```
ubuntu@thinknyxlinux:/tmp$ ls -lh
total 24K
drwx----- 2 root root 4.0K Nov 22 08:44 snap-private-tmp
drwx----- 3 root root 4.0K Nov 22 08:45 systemd-private-311f0da5651842b38945cabffffd9098-ModemManager.service-qXDUSo
drwx----- 3 root root 4.0K Nov 22 08:45 systemd-private-311f0da5651842b38945cabffffd9098-chrony.service-9qEvUa
drwx----- 3 root root 4.0K Nov 22 08:45 systemd-private-311f0da5651842b38945cabffffd9098-polkit.service-TZQXIv
drwx----- 3 root root 4.0K Nov 22 08:45 systemd-private-311f0da5651842b38945cabffffd9098-systemd-logind.service-Ep9JBB
drwx----- 3 root root 4.0K Nov 22 08:44 systemd-private-311f0da5651842b38945cabffffd9098-systemd-resolved.service-E4Gr0u
```

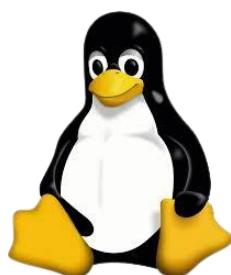
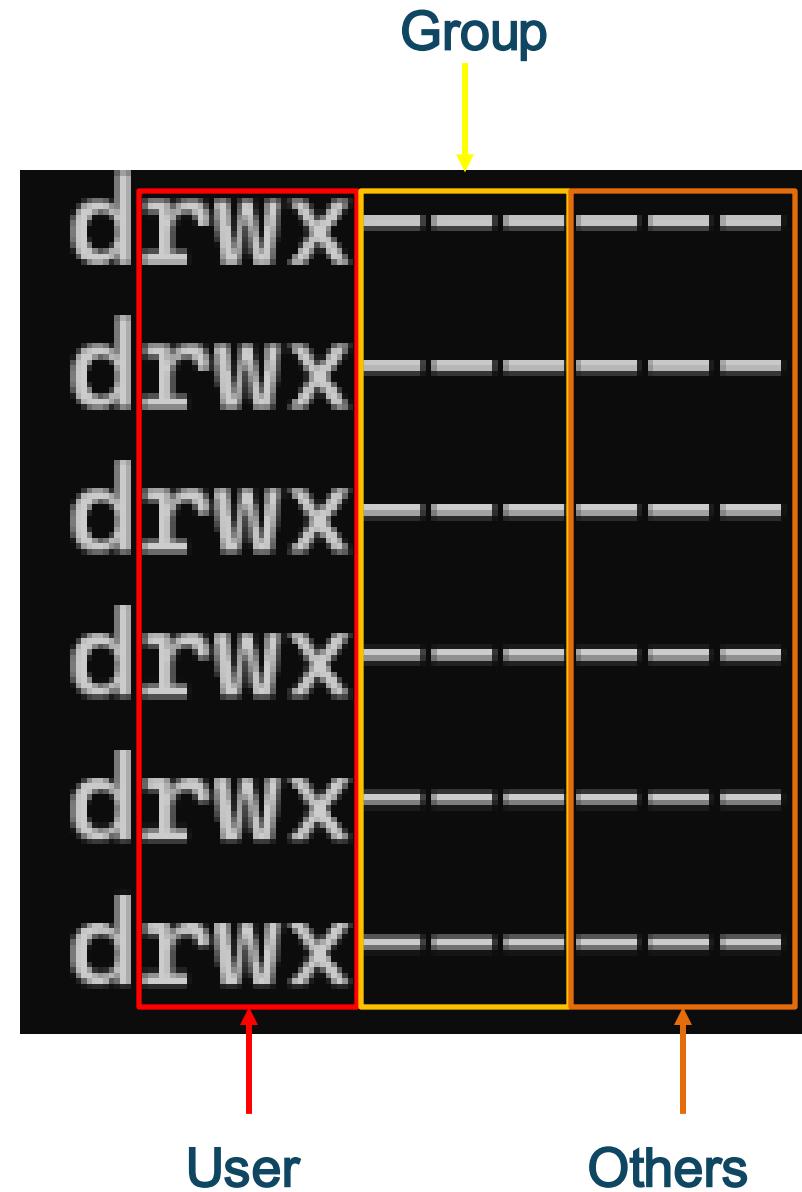
Permissions



File Ownership

➤ File permissions

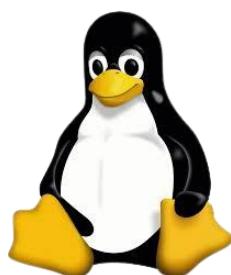
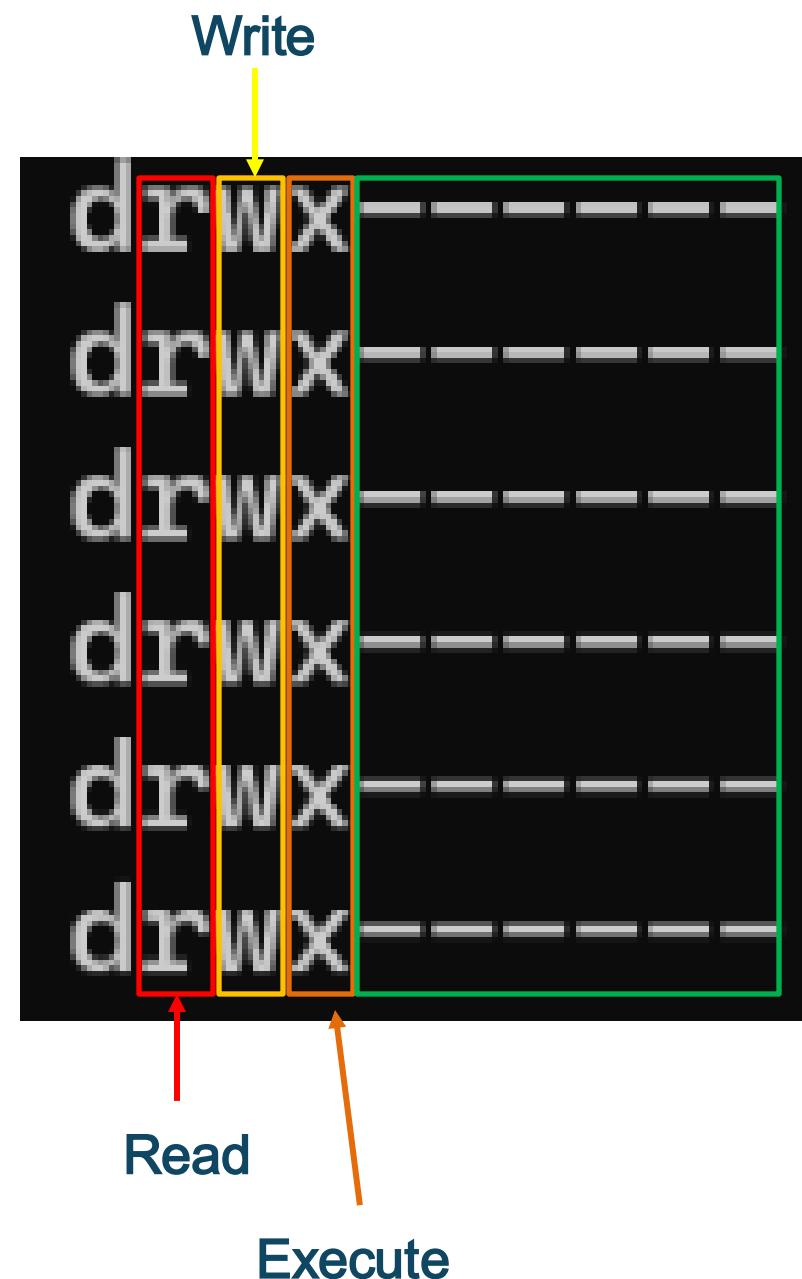
- ✓ User : Owner of the file
- ✓ Group : Set of users
- ✓ Others : Everyone else



File Ownership

➤ File permissions

- ✓ User : Owner of the file
- ✓ Group : Set of users
- ✓ Others : Everyone else



File Ownership

➤ File permission management commands

Command	Description	Syntax	Usage
chmod	Changes file mode bits	chmod [options] <mode> <file>	chmod u+rx thinknyx.txt
chown	Changes file owner and group	chown [options] <owner[:group]> <file>	chown alex:developers filename
chgrp	Changes group ownership	chgrp [options] <group> <file>	chgrp developers thinknyx.txt



File Ownership

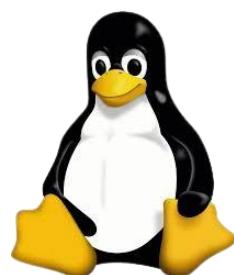
➤ Different ways to use chmod

□ Symbolic and Octal or Numeric Method

Symbolic Method

Permission groups

u	user
g	group
o	others
a	all



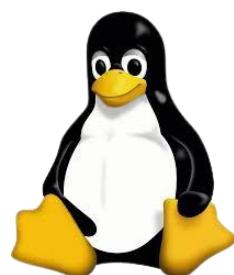
File Ownership

- Different ways to use chmod
 - Symbolic and Octal or Numeric Method

Symbolic Method

Permission types

r	read permission
w	write permission
x	execute permission



File Ownership

➤ Different ways to use chmod

□ Symbolic and Octal or Numeric Method

Symbolic Method

Operators

+	add permission
-	remove permission
=	set permission explicitly, replacing others



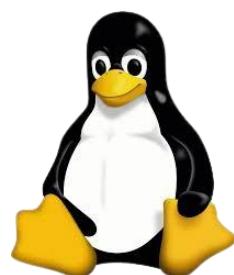
File Ownership

- Different ways to use chmod

```
chmod u+x thinknyx.txt
```

```
chmod g-r thinknyx.txt
```

```
chmod u=rw,g=r,o=r thinknyx.txt
```



File Ownership

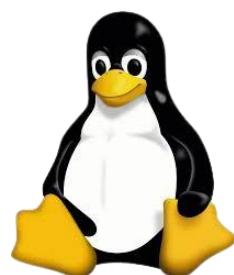
➤ Different ways to use chmod

□ Symbolic and Octal or Numeric Method

Octal Method

Permission

4	read permission (r)
2	write permission (w)
1	execute permission (x)

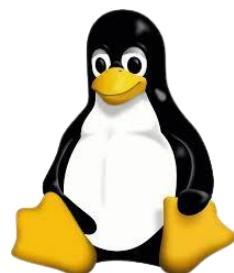


File Ownership

- Different ways to use chmod
 - Symbolic and Octal or Numeric Method

Octal Method

Octal Representation				
0	000	- - -	-	No permissions
1	001	- - x	-	Only Execute
2	010	- w -	-	Only Write
3	011	- w x	-	Write and Execute
4	100	r - -	-	Only Read
5	101	r - x	-	Read and Execute
6	110	r w -	-	Read and Write
7	111	r w x	-	Read, Write and Execute



File Ownership

➤ Numeric value

- ✓ 7 means Full access: Read (4) + Write (2) + Execute (1)
- ✓ 6 means Read and Write: Read (4) + Write (2)
- ✓ 5 means Read and Execute: Read (4) + Execute (1)

```
chmod 765 thinknyx.txt
```

Binary Conversion

$$\begin{array}{r} 1 \\ - 4 \\ \hline 1 \\ \quad - 2 \\ \hline 1 \end{array}$$

$$4 + 2 + 1 = 7$$

(u) user

1	1	1
r	w	x
<hr/>		
7		

(g) group

1	1	0
r	w	x
<hr/>		
6		

(g) group

1	0	1
r	w	x
<hr/>		
5		



File Ownership

➤ Effects of Permissions on Files and Directories

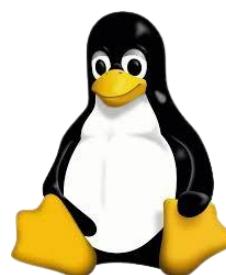
Permissions	File Permissions	Directory Permissions
Read (r)	Allows the user to viewing the contents of the file	Allows listing the directory's contents
Write (w)	Allows the user to modifying the contents of the file	Allows creating or deleting files within the directory
Execute (x)	Allows the user to executing the file as a program or command	Allows traversing the directory and accessing files and subdirectories inside the directory



File Ownership

➤ Special Cases of Directory Permissions

- ✓ **Read & Execute:** User can list and access file contents in read-only mode but cannot modify files
- ✓ **Read-Only:** User can list file names but cannot view permissions or timestamps
- ✓ **Execute-Only:** User can cd into the directory but cannot list file names; file access requires exact names



File Ownership

➤ Directory Ownership and File Deletion

- ✓ **Write Permissions:** Directory owners or users with write access can delete files inside, regardless of file ownership or permissions
- ✓ **Sticky Bit:** Restricts file deletion to file owners, even if others have write access to the directory





Demonstration | File Ownership



Linux | Introduction to Packages



Introduction to Packages



Introduction to Packages

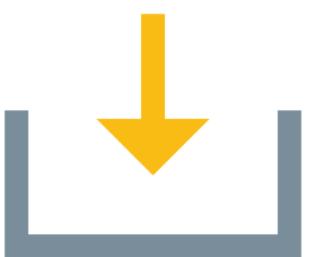
➤ What is Package



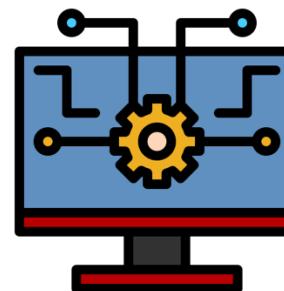
Install Software

Includes:

- ✓ Binaries
- ✓ Libraries
- ✓ Configuration files
- ✓ Documentation



Install



Manage Software



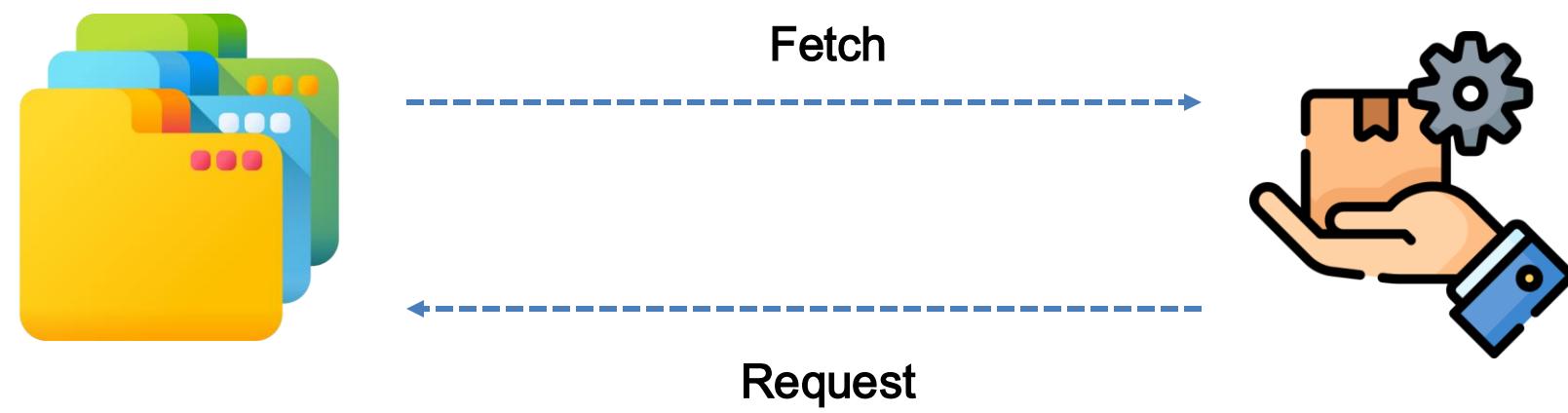
Update



Repositories



Repositories



- ✓ Online stores for software
- ✓ Contain precompiled software packages
- ✓ Ready to be downloaded
- ✓ Installed with a command

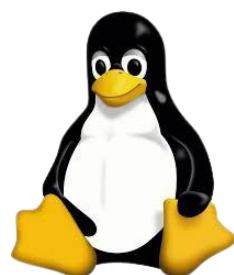


Repositories

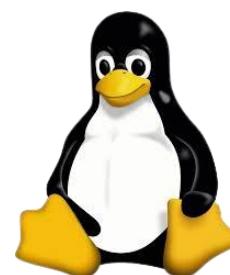
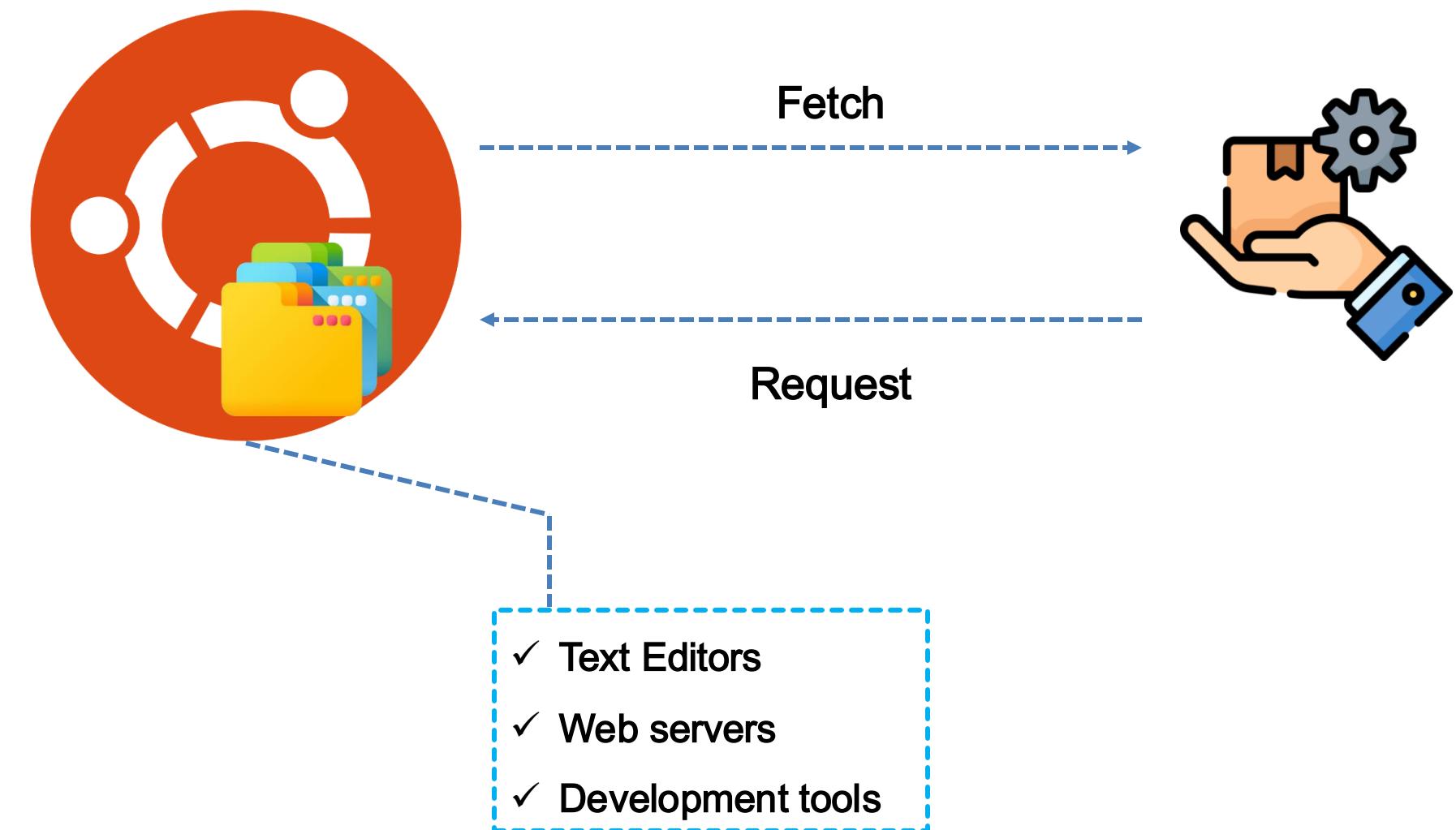


Meets different criteria like:

- ✓ **Main Repositories**
- ✓ **Security Updates**
- ✓ **Third-Party software Repositories**



Managing Packages

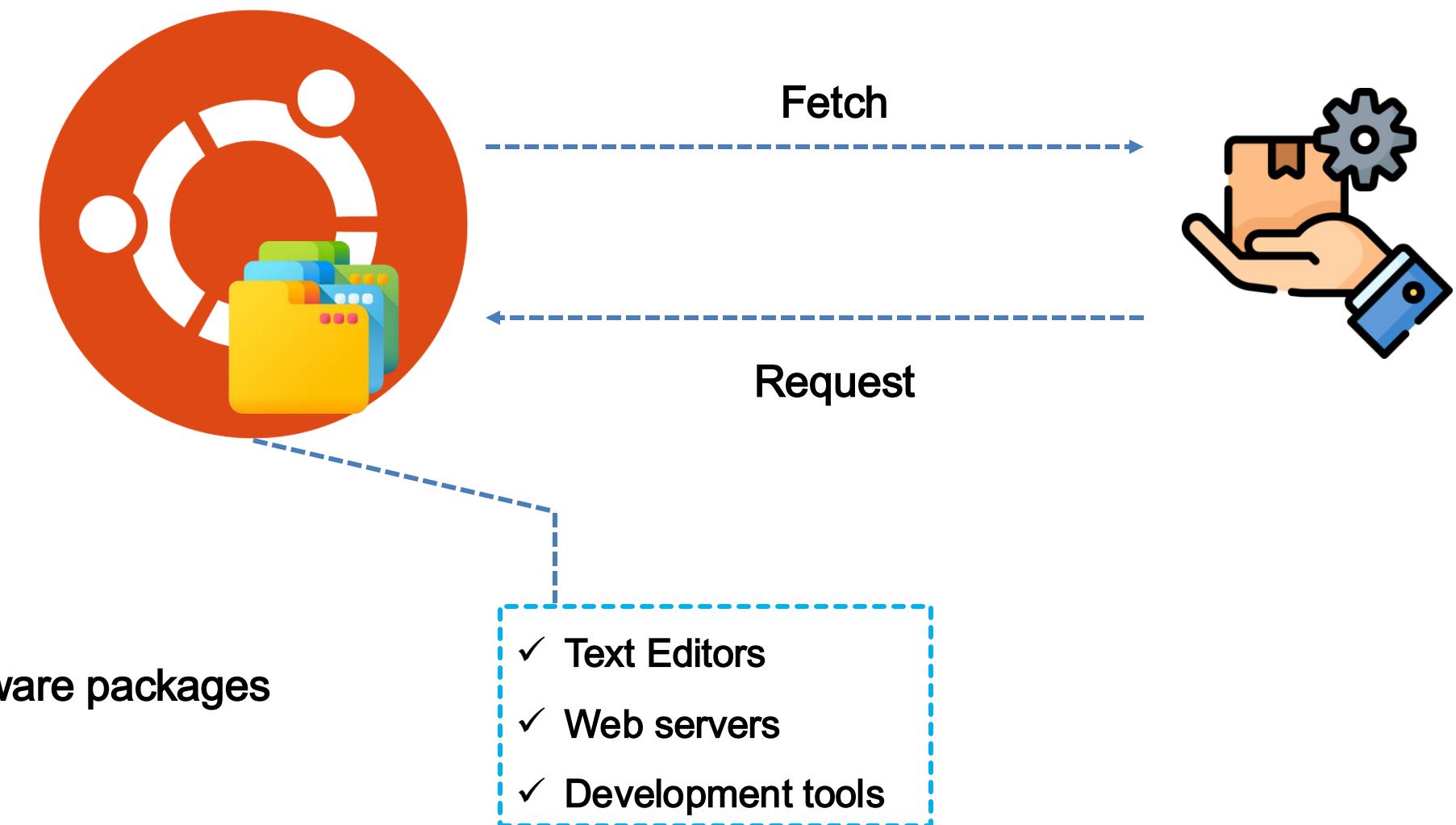


Managing Packages

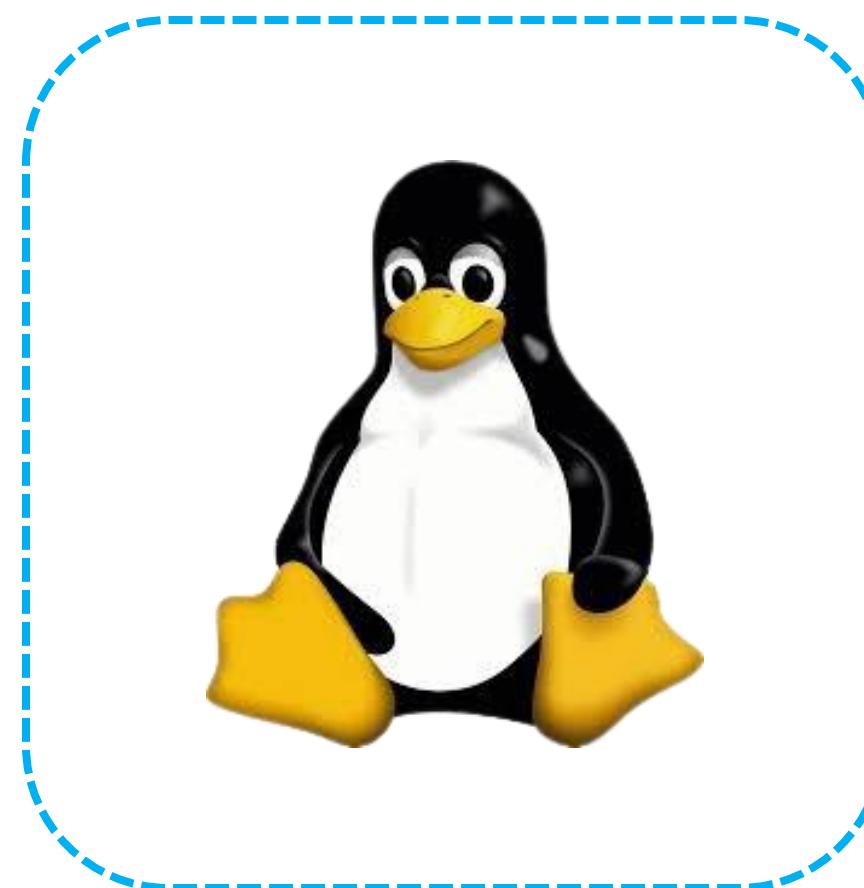
➤ Repositories



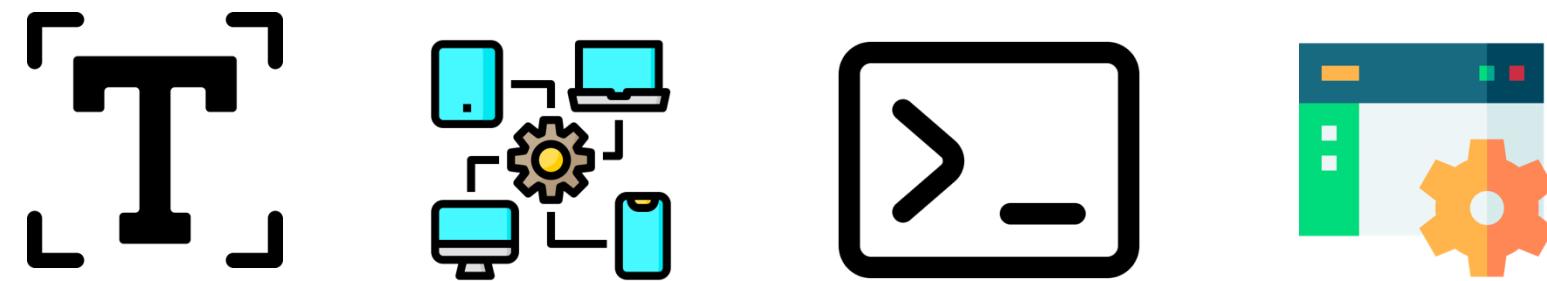
- ✓ Online stores for software
- ✓ Contain precompiled software packages
- ✓ Ready to be downloaded
- ✓ Installed with a command



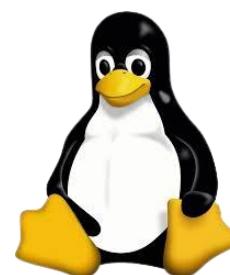
Managing Packages



Everything is a file



- ✓ Simplifies system management
- ✓ Allows uniformity in how components are accessed and manipulated



Common Methods of software installation

➤ Binary Copy

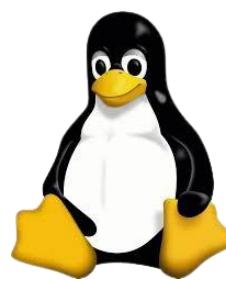
- ✓ Precompiled binaries
- ✓ Ready-to-run executables
- ✓ Offer quick installation
- ✓ Ideal for users who don't need to modify software



Common Methods of software installation

➤ Source Code

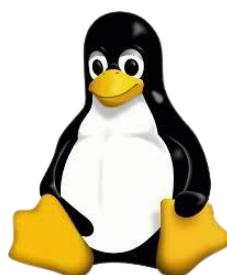
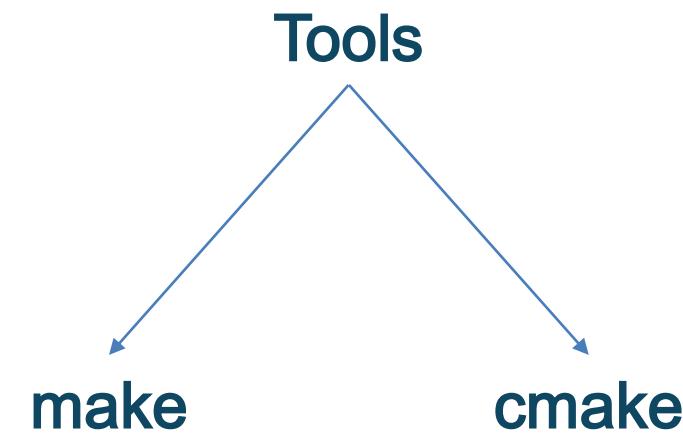
- ✓ Raw programming code
- ✓ Written before its compiled into an executable file



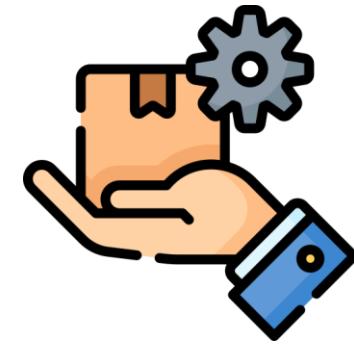
Common Methods of software installation

➤ Self-Compiling Packages

- ✓ Require manual assurance
- ✓ All libraries and tools (dependencies) are installed



Package Manager



Bundled archives

- ✓ Precompiled binary files
- ✓ Necessary installation scripts
- ✓ Configuration files
- ✓ Metadata



Install

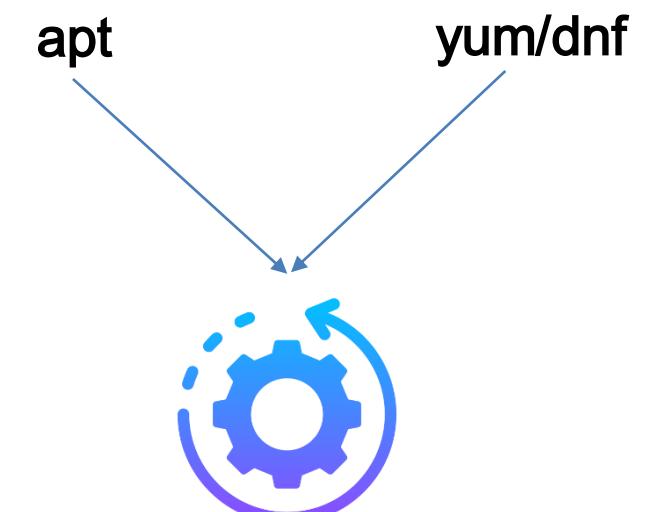


Upgrading



Removing

- ✓ Provide components needed to install and manage software efficiency
- ✓ Simplify process by automating resolution and installation

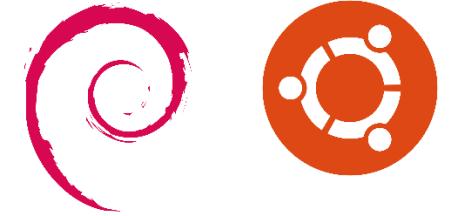


Package Terminologies

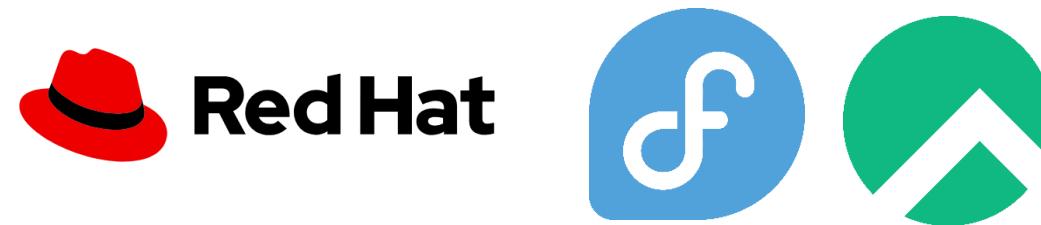
➤ Package Formats

- ✓ Comes in different format
- ✓ Depends upon linux distribution
- ✓ Defines how software is packaged, distributed and installed

.deb



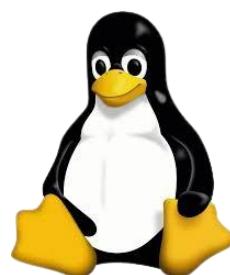
.rpm



Package Terminologies

➤ Package Manager

- ✓ Helps in automate installations, maintenance and removal of package



Package Terminologies

➤ Repository

- ✓ Centralized location containing software packages
- ✓ Packages are tested and optimized for specific linux distributions

Debian Based

“/etc/apt/source.list”

Red Hat Based

“/etc/yum.repos.d”



Package Terminologies

➤ Dependencies

- ✓ Some packages require other packages to function correctly
- ✓ Tools like – apt, yum and dnf automatically handle these requirements



Package Terminologies

➤ Open Source

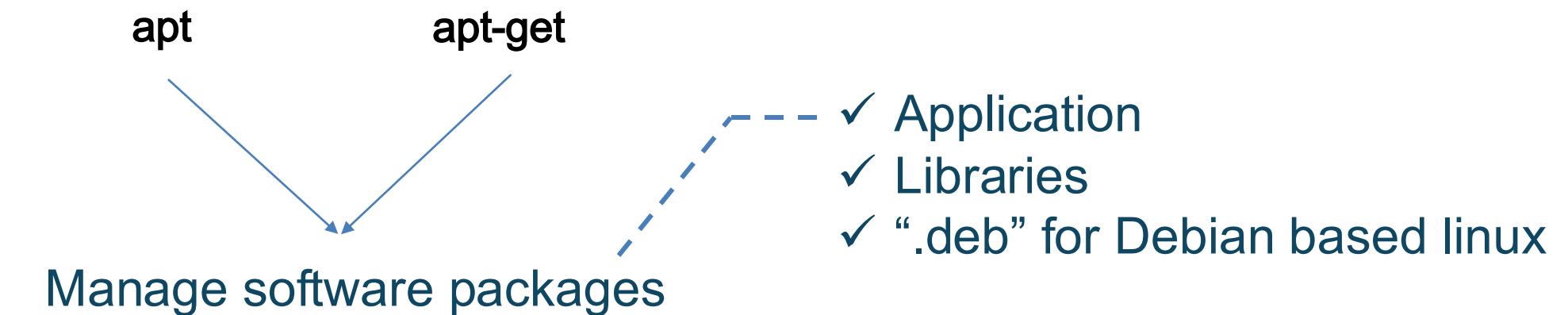
- ✓ Most distributions offer open-source software packages
- ✓ Integrate seamlessly with system
- ✓ Source code can be downloaded from project website



Package Management Tools

➤ apt

- ✓ Advance Package Tool



apt:

- ✓ User-friendly, High-level interface
- ✓ Intuitive commands
- ✓ Enhanced security features
- ✓ Predictable behavior

apt-get:

- ✓ Lower-level tool
- ✓ Interacts more closely with core linux processes



Package Management Tools

➤ yum

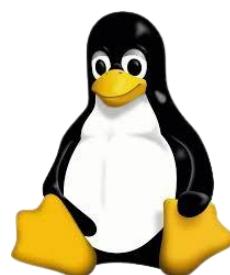
- ✓ Yellowdog Updater Modified
- ✓ Package manager for “.rpm” packages on Red-Hat based distribution



Package Management Tools

➤ dnf

- ✓ Dandified YUM
- ✓ Modernized replacement for yum



Package Management Tools

➤ **zypper**

- ✓ Command-line package manager for OpenSUSE and SUSE Linux Enterprise
- ✓ Advanced dependency resolution and repository management





Linux | apt Package Manager



apt Package Manager

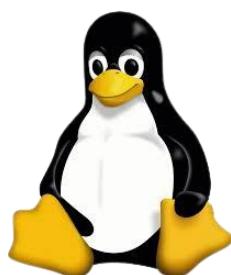
A deb package file name is structured as:

name_version+release_architecture.deb

Example

vim_8.2.2434-3+deb11u1_amd64.deb

- ✓ **NAME:** Describes the contents of the package (e.g., vim)
- ✓ **VERSION:** The software's version number (8.2.2434-3)
- ✓ **RELEASE:** The release number set by the packager (deb11u1)
- ✓ **ARCHITECTURE:** The CPU architecture the package is compiled for:
 - amd_64: 64-bit systems
 - aarch64 for ARM processors



apt Package Manager

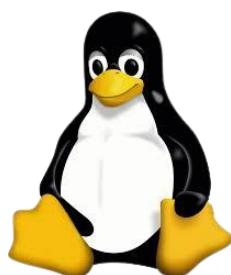
A deb package file name is structured as:

name_version+release_architecture.deb

Example

vim_8.2.2434-3+deb11u1_amd64.deb

- ✓ **NAME:** Describes the contents of the package (e.g., vim)
- ✓ **VERSION:** The software's version number (8.2.2434-3)
- ✓ **RELEASE:** The release number set by the packager (deb11u1)
- ✓ **ARCHITECTURE:** The CPU architecture the package is compiled for:
 - amd_64: 64-bit systems
 - aarch64 for ARM processors



apt Package Manager

- ✓ dpkg focuses on individual .deb files
 - ✓ apt acts as a user-friendly front end, streamlining tasks
 - Searching repositories
 - Resolving dependencies
 - Handling software updates
- Advanced Package Tool (apt)
- ✓ Search for available software in the repositories
 - ✓ Install or upgrade packages along with their dependencies
 - ✓ Keep your system secure and up-to-date with regular updates and upgrades
 - ✓ Remove packages or free up disk space by clearing the local cache



apt Package Manager

Command	Purpose	Example
<code>sudo apt update</code>	Refreshes & updates the local package cache with the latest repository information	<code>sudo apt update</code>
<code>sudo apt upgrade</code>	Upgrades all installed packages to their latest versions	<code>sudo apt upgrade</code>
<code>sudo apt full-upgrade</code>	Installs updates, handling changes in dependencies	<code>sudo apt full-upgrade</code>
<code>sudo apt search <package-name></code>	Searches for a specific package in the repository	<code>sudo apt search vim</code>
<code>sudo apt install <package-name></code>	Installs a specified package along with its dependencies	<code>sudo apt install vim</code>



apt Package Manager

Command	Purpose	Example
<code>dpkg -l</code>	List of installed packages with details like name, version, and architecture	<code>dpkg -l</code>
<code>sudo apt remove</code>	Removes a specified package but keeps its configuration files	<code>sudo apt remove vim</code>
<code>sudo apt purge</code>	Installs updates, handling changes in dependencies	<code>sudo apt purge vim</code>
<code>sudo apt clean</code>	Clears the local repository cache, freeing up disk space	<code>sudo apt clean</code>





Linux | Process Management



Process Management

➤ Fundamental Commands Related to Processes

Command	Description	Syntax	Usage
pidof	Finds the process ID of a running program	pidof [options] <program_name>	pidof sshd
ps	Displays information about a selection of the active processes	ps [options]	ps aux
pstree	Shows running processes as a tree	pstree [options]	pstree
pgrep	Searches for processes by name and returns their process IDs	pgrep [options] <pattern>	pgrep -u user1



a	all users
u	user, CPU, and memory usage
x	not tied to a specific terminal



Process Management

➤ Kill Command



Process Management

➤ Kill Command

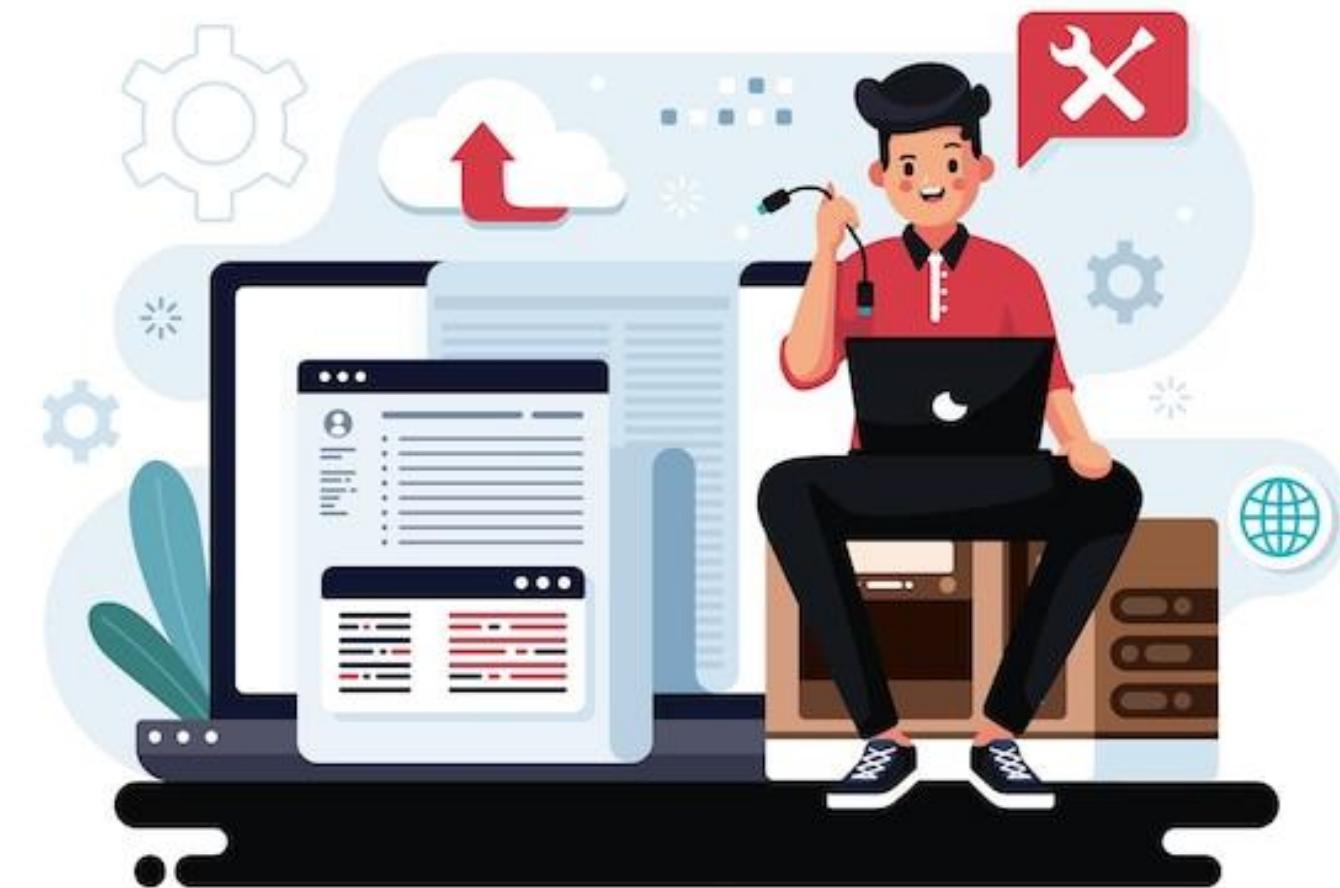
- ✓ Terminate the Process
- ✓ Free Up Resources
- ✓ Restore Server Performance
- ✓ Maintain User Experience



Process Management

➤ Kill Command

- ✓ Sends signals to manage processes, such as TERM, KILL, or STOP
- ✓ Default signal: TERM (graceful termination)
- ✓ KILL: Forcefully stops a process
- ✓ STOP: Pauses the process



Process Management

➤ Kill Command

- ✓ Only owners or root users can kill processes

- ✓ The term "kill" can be misleading
- ✓ The command sends various signals, not limited to termination



Process Management

1) SIGHUP	2) SIGINT	3) SIGQUIT	4) SIGILL	5) SIGTRAP
6) SIGABRT	7) SIGBUS	8) SIGFPE	9) SIGKILL	10) SIGUSR1
11) SIGSEGV	12) SIGUSR2	13) SIGPIPE	14) SIGALRM	15) SIGTERM
16) SIGSTKFLT	17) SIGCHLD	18) SIGCONT	19) SIGSTOP	20) SIGTSTP
21) SIGTTIN	22) SIGTTOU	23) SIGURG	24) SIGXCPU	25) SIGXFSZ
26) SIGVTALRM	27) SIGPROF	28) SIGWINCH	29) SIGIO	30) SIGPWR
31) SIGSYS	34) SIGRTMIN	35) SIGRTMIN+1	36) SIGRTMIN+2	37) SIGRTMIN+3
38) SIGRTMIN+4	39) SIGRTMIN+5	40) SIGRTMIN+6	41) SIGRTMIN+7	42) SIGRTMIN+8
43) SIGRTMIN+9	44) SIGRTMIN+10	45) SIGRTMIN+11	46) SIGRTMIN+12	47) SIGRTMIN+13
48) SIGRTMIN+14	49) SIGRTMIN+15	50) SIGRTMAX-14	51) SIGRTMAX-13	52) SIGRTMAX-12
53) SIGRTMAX-11	54) SIGRTMAX-10	55) SIGRTMAX-9	56) SIGRTMAX-8	57) SIGRTMAX-7
58) SIGRTMAX-6	59) SIGRTMAX-5	60) SIGRTMAX-4	61) SIGRTMAX-3	62) SIGRTMAX-2
63) SIGRTMAX-1	64) SIGRTMAX			



Process Management

➤ Syntax for the kill command

`kill [options] <pid>`



Signal number (e.g., 9 for SIGKILL)

Full signal name (e.g., SIGKILL)

Signal name without the SIG prefix (e.g., KILL)

```
kill -9 <pid>
kill -SIGKILL <pid>
kill -KILL <pid>
```



Process Management

PID Value	Effect
PID > 0	Signal targets the process with the specified PID
PID = 0	Signal targets all processes in the current process group
PID = -1	Signal targets all user-owned processes; as root, it excludes init and itself
PID < -1	Signal targets all processes in the group with a GID matching the absolute value of the PID

- ✓ Use signal names, not numbers, for consistency and cross-platform portability



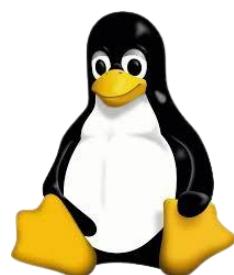
Process Management

killall

- ✓ Unsure of the process ID or prefer to target processes by name

pkill

- ✓ Sends signals to processes based on names, more flexible than killall
- ✓ Uses pgrep criteria (name, user, pattern) to target processes



Process Management

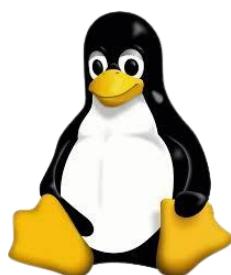
For example

```
pkill -u ubuntu vi
```

Default signal (SIGTERM) to all matching processes

```
pkill -9 -u ubuntu vi
```

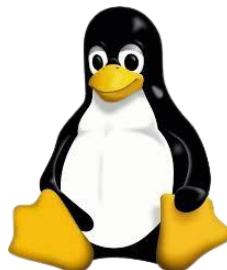
Additional options -> terminal (-t) or session (-s)



Process Management

➤ SIGTERM vs SIGKILL

- ✓ Avoid killing system processes to prevent instability or crashes
- ✓ SIGTERM: Gracefully terminates processes
- ✓ SIGKILL: Forcefully terminates without cleanup, causing potential issues
- ✓ Use signals carefully to avoid unintended consequences





Demonstration | Managing Services

User Management

File	Description
UNIT	Service unit name, e.g., cron.service
LOAD	Loaded into memory; typically shows loaded if successful
ACTIVE	Common states include active or inactive
SUB	Detailed state, varies by service type and behavior, e.g., running, exited, waiting (active), or dead, failed (inactive)
Description	Brief purpose of the service



User Management

Active State	Description
active (exited)	Service started successfully and completed its task (one-shot)
active (waiting)	Service running but waiting for an event or condition to proceed





Demonstration | Deploy the Python app to EC2

Summary



- ❖ Introduction to terminal and basic commands: echo, uname, whoami, who, uptime, clear, history, man, whatis, whereis, which



Summary



- ❖ **Files & directories:** relative vs absolute paths, pwd, cd, ls, mkdir, rmdir
- ❖ **File operations:** touch, file, cp, mv, rm
- ❖ **Viewing file contents:** cat, tac, head, tail, more, less



Summary



- ❖ **Text editors:** nano, vi
- ❖ **File ownership & permissions:** users, groups, others; chmod, chown, chgrp
(symbolic & octal)



Summary



-
- ❖ **Packages & repositories:** apt commands (update, upgrade, full-upgrade, search, install, dpkg, remove, purge, clean)
 - ❖ **Processes:** pidof, ps, pstree, pgrep
-



Summary



- ❖ Services management: systemctl
- ❖ AWS EC2: launching instance, deploying Python, activating virtual environments (two methods)





Demonstration | Updating Linux Progress in GitHub Projects



The Ultimate Linux Bootcamp for DevOps SRE & Cloud Engineers



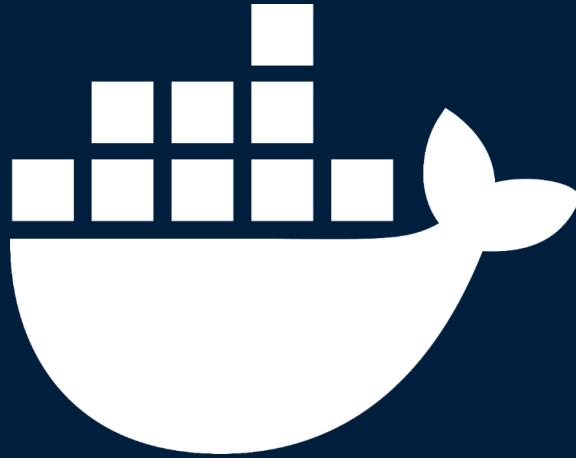
Thank You





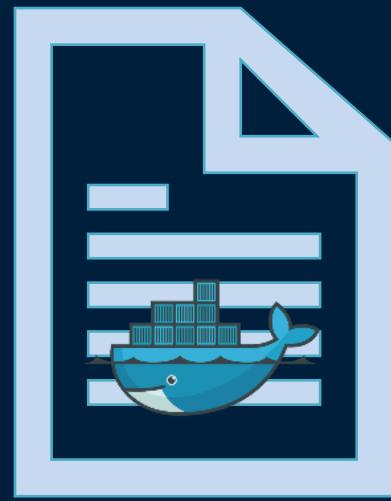
Docker | Overview of Docker Section

Overview of Docker Section

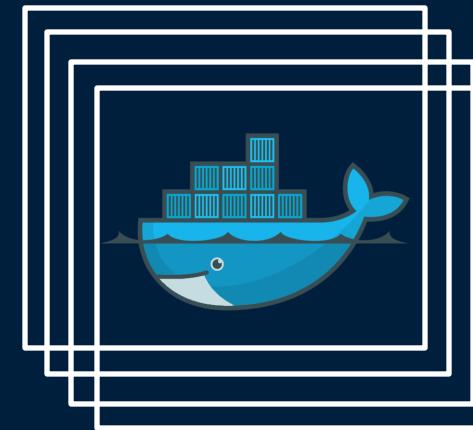


- Docker as an essential tool in modern DevOps
- Packages applications with dependencies into containers
- Ensures consistency across environments (**laptop → staging → production**)
- Streamlines deployment process
- Improves scalability of applications
- Enhances collaboration between development and operations teams

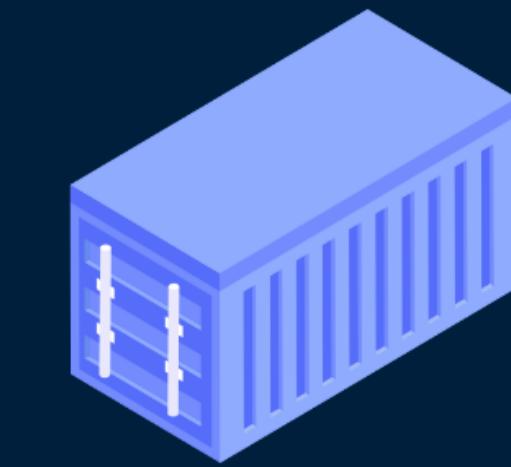
Overview of Docker Section



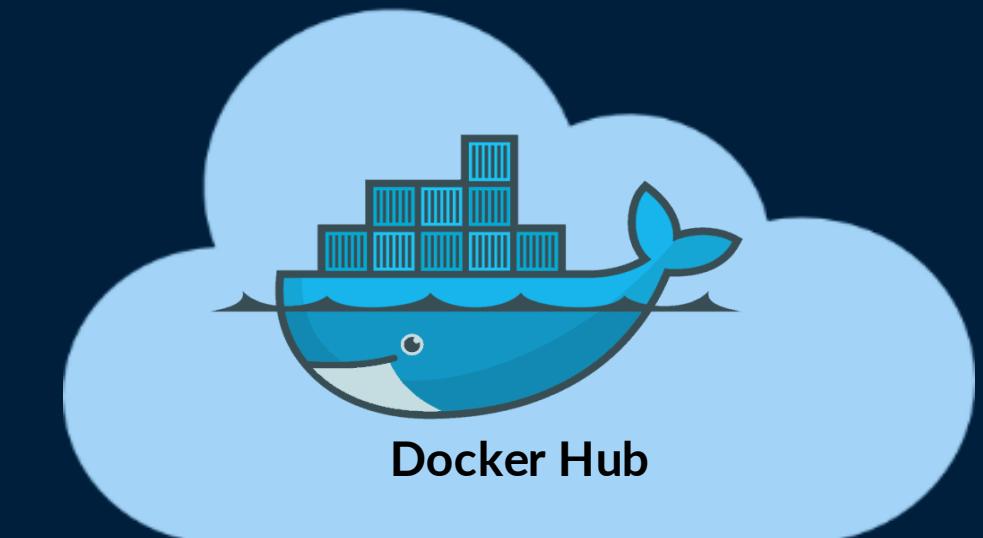
Dockerfile



Image



Container



Introduction to Containerization



Introduction to Containerization

Advantages of Containers



Environment consistency



Faster scalability



Development speed

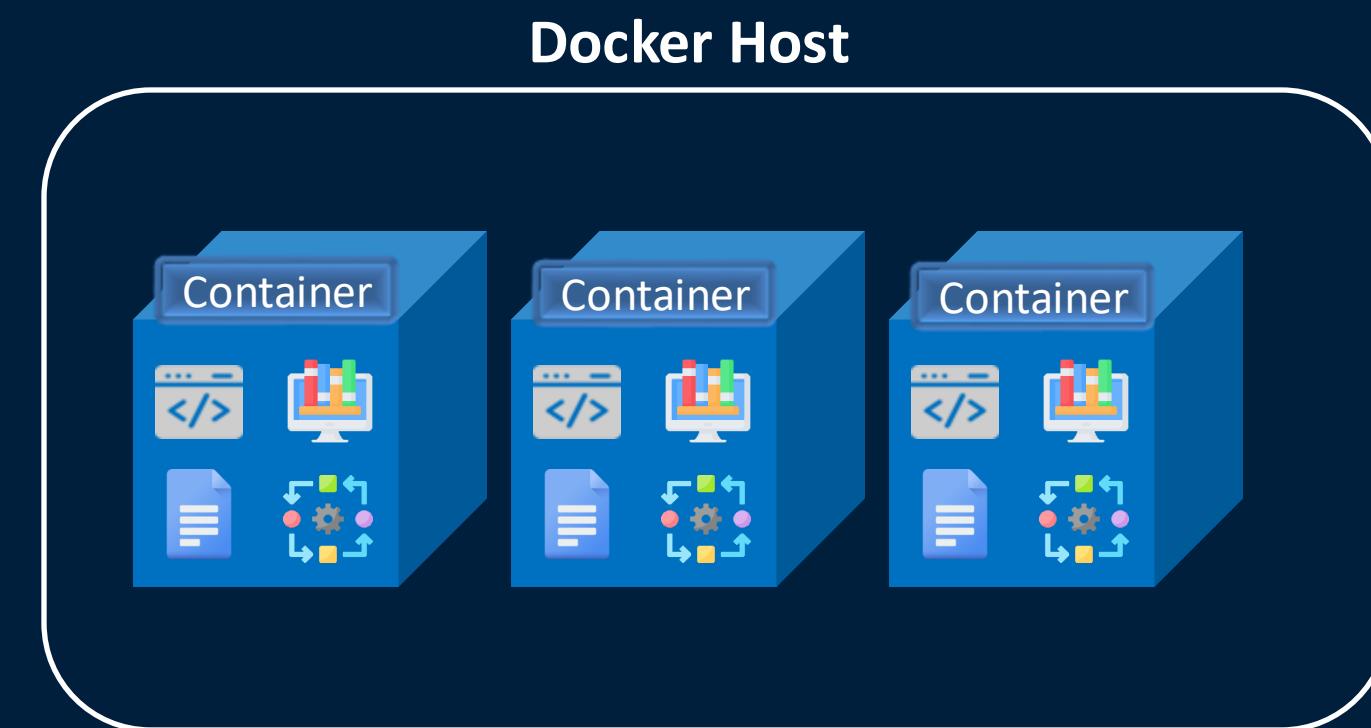


Quick startup

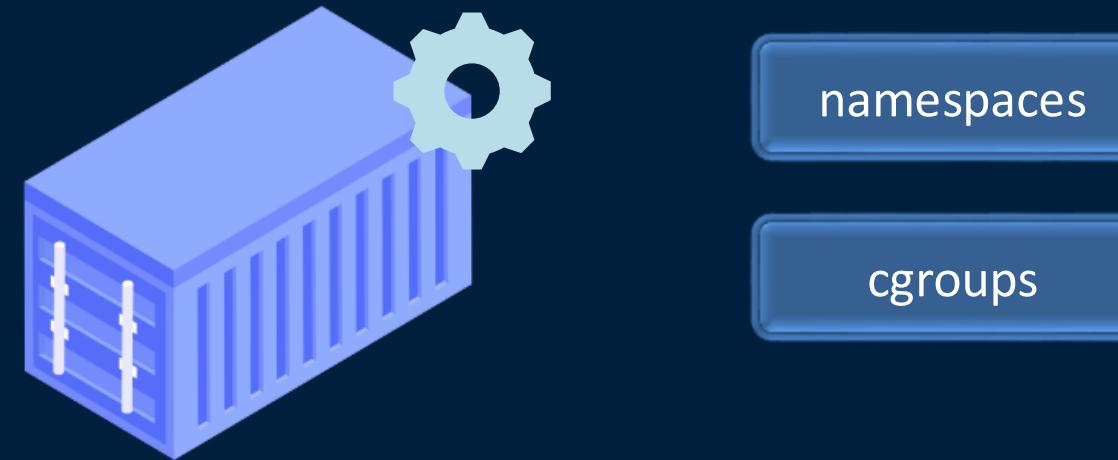


Portability

Containerization Concepts



Containerization Concepts



Containerization Concepts

namespaces



namespaces

- ✓ Isolate and virtualize system resources
- ✓ Ensure each process accesses unique set of resources

Containerization Concepts

namespaces

namespaces

USER

PID

MNT

IPC

UTS

NET

Containerization Concepts

cgroups



Control group

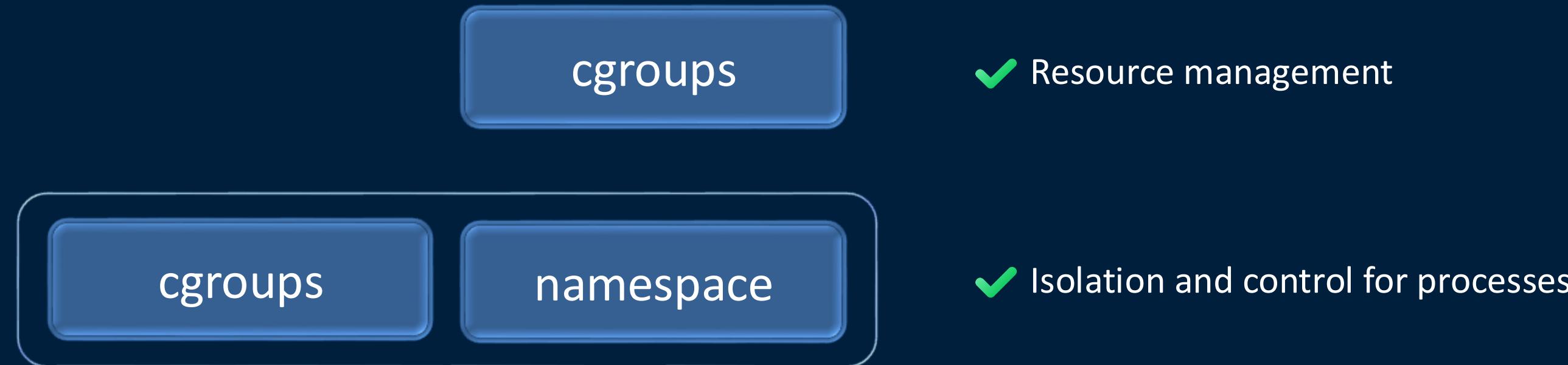
cgroups

- ✓ Isolates the resource usage such as CPU, memory, disk I/O, network etc



Containerization Concepts

cgroups





Containerization Concepts

Eric
Biederman

cgroups

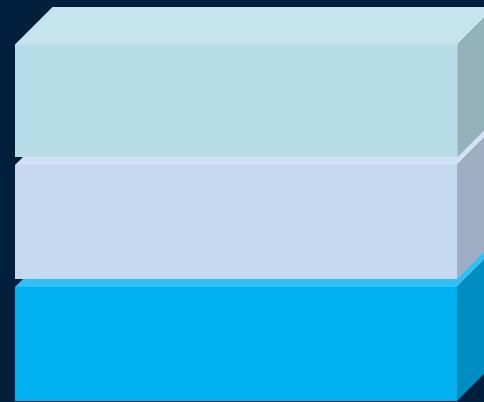
namespace

Paul Menage &
Rohit Seth

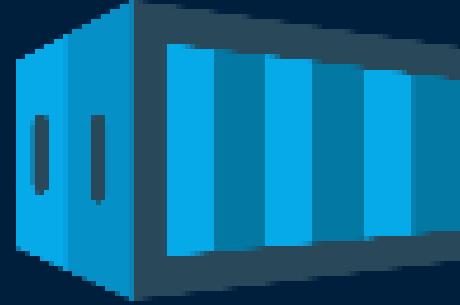
Docker at a Glance



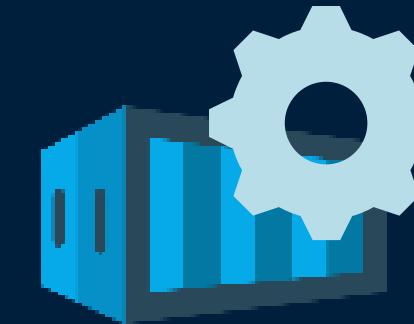
- ✓ Decouples applications
- ✓ Accelerating code deployment to all environments



Build



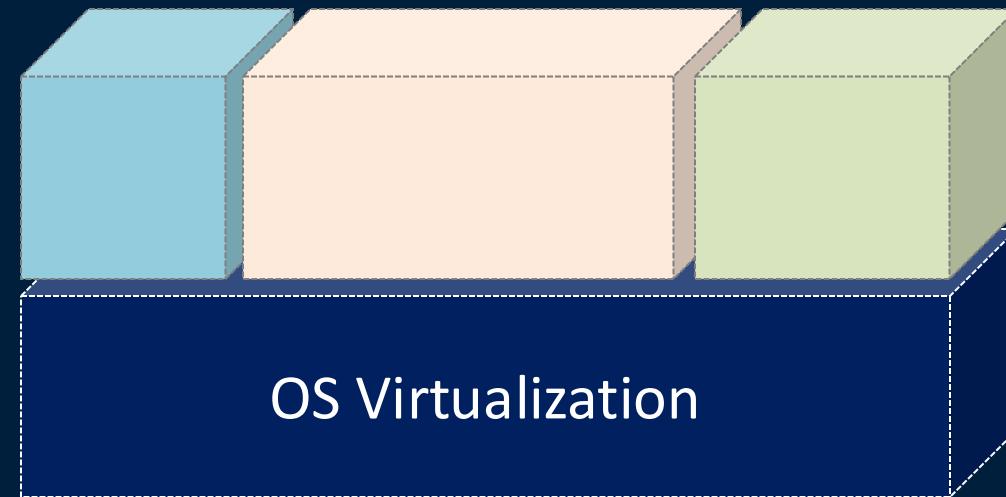
Ship



Run

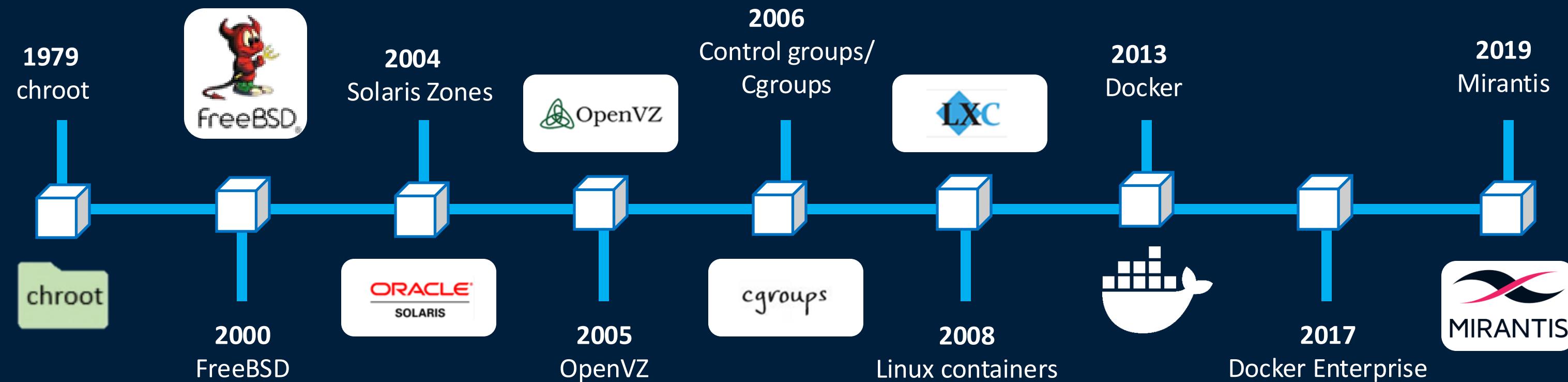
Docker at a Glance

History of Containers & Docker



Docker at a Glance

History of Containers & Docker

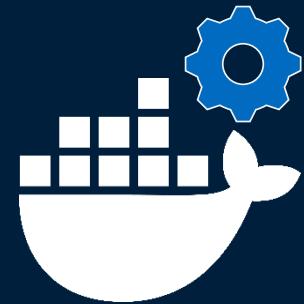


Docker at a Glance

Docker Products



Docker
Desktop



Docker
Engine



Docker
Hub

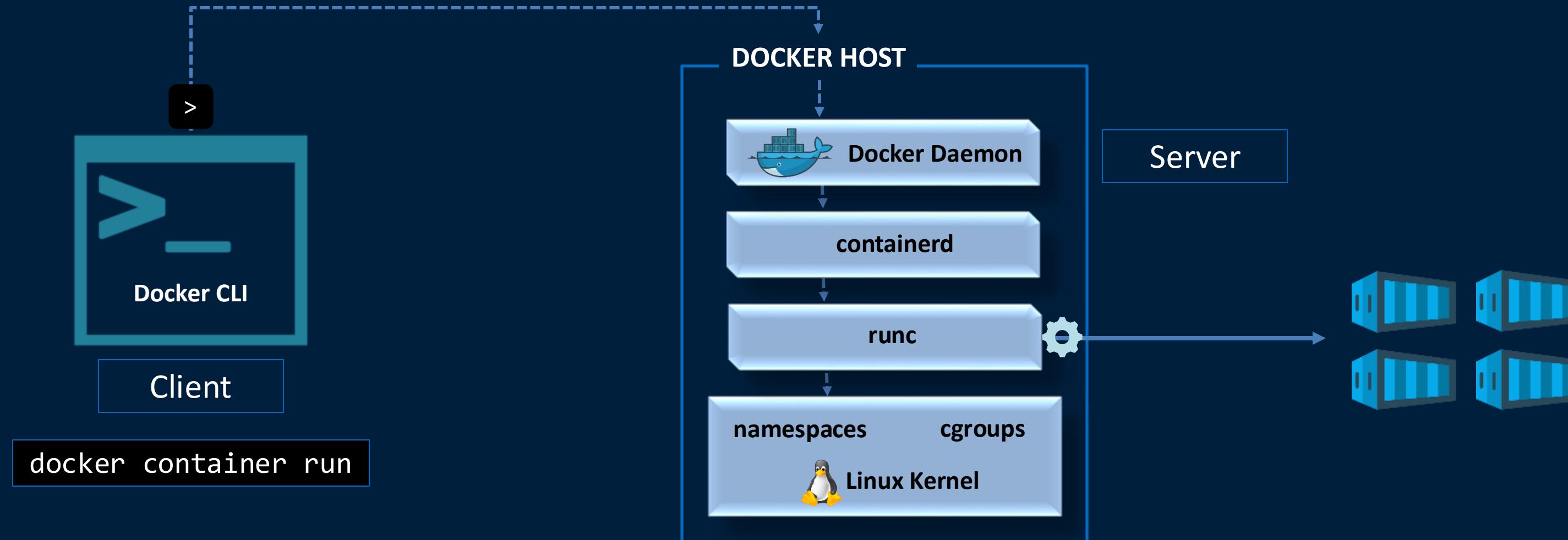


Docker
Scout

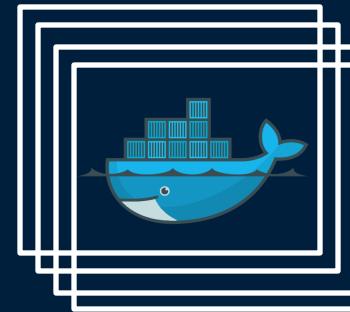


Docker Build
Cloud

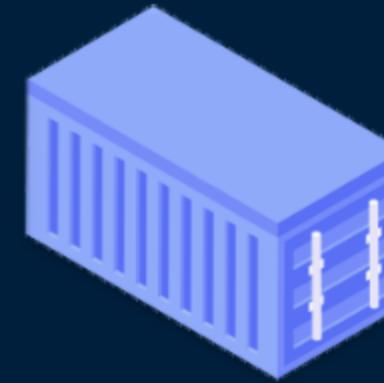
Docker Architecture



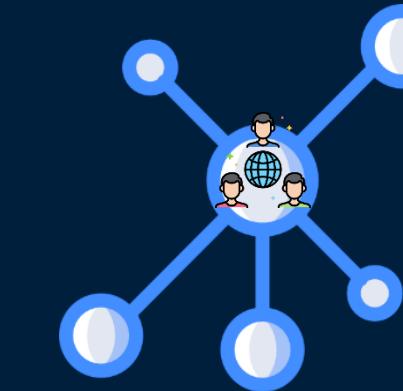
Docker Objects



Images



Containers



Networks



Volumes

Docker Objects

Images

Read-only templates
Set of instructions



Docker Objects

Containers



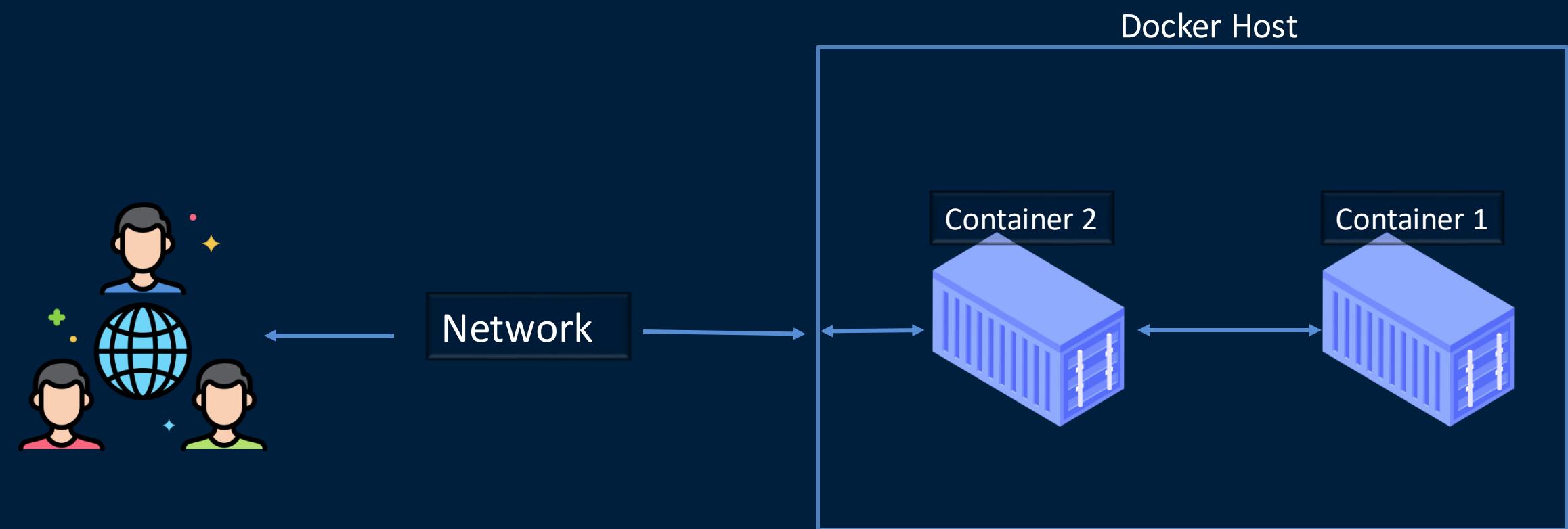
Lifecycle

- Create
- Start
- Stop

Docker Objects

Networks

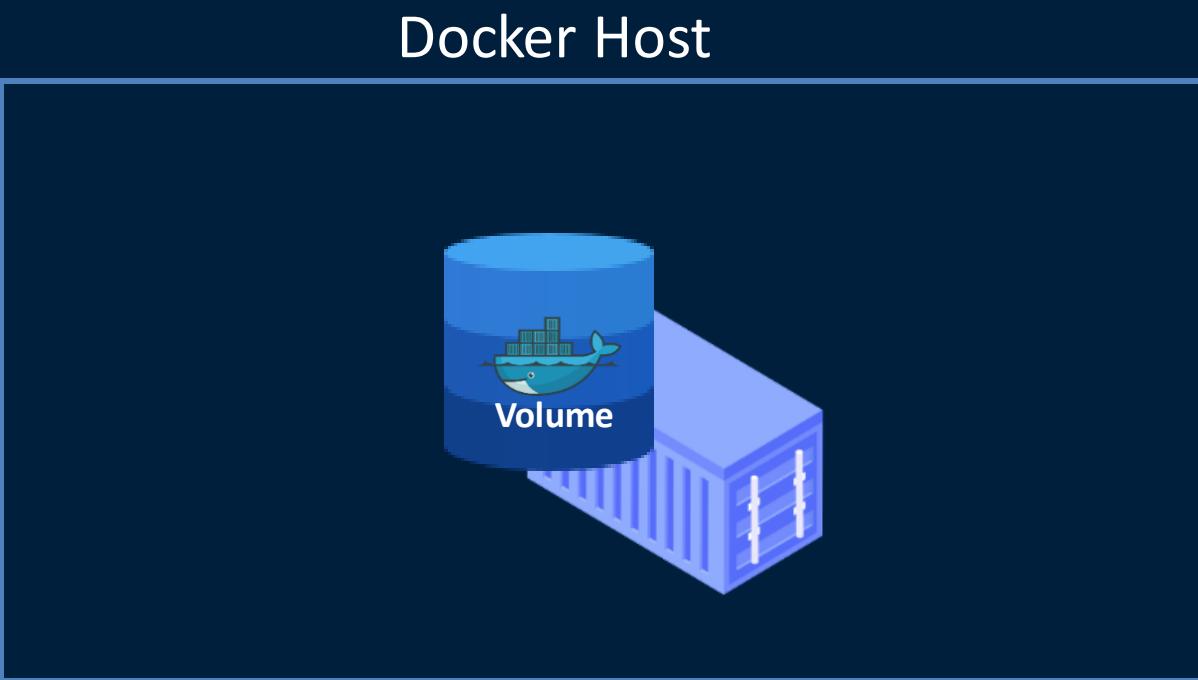
Default network
User-defined networks



Docker Objects

Volumes

Ephemeral

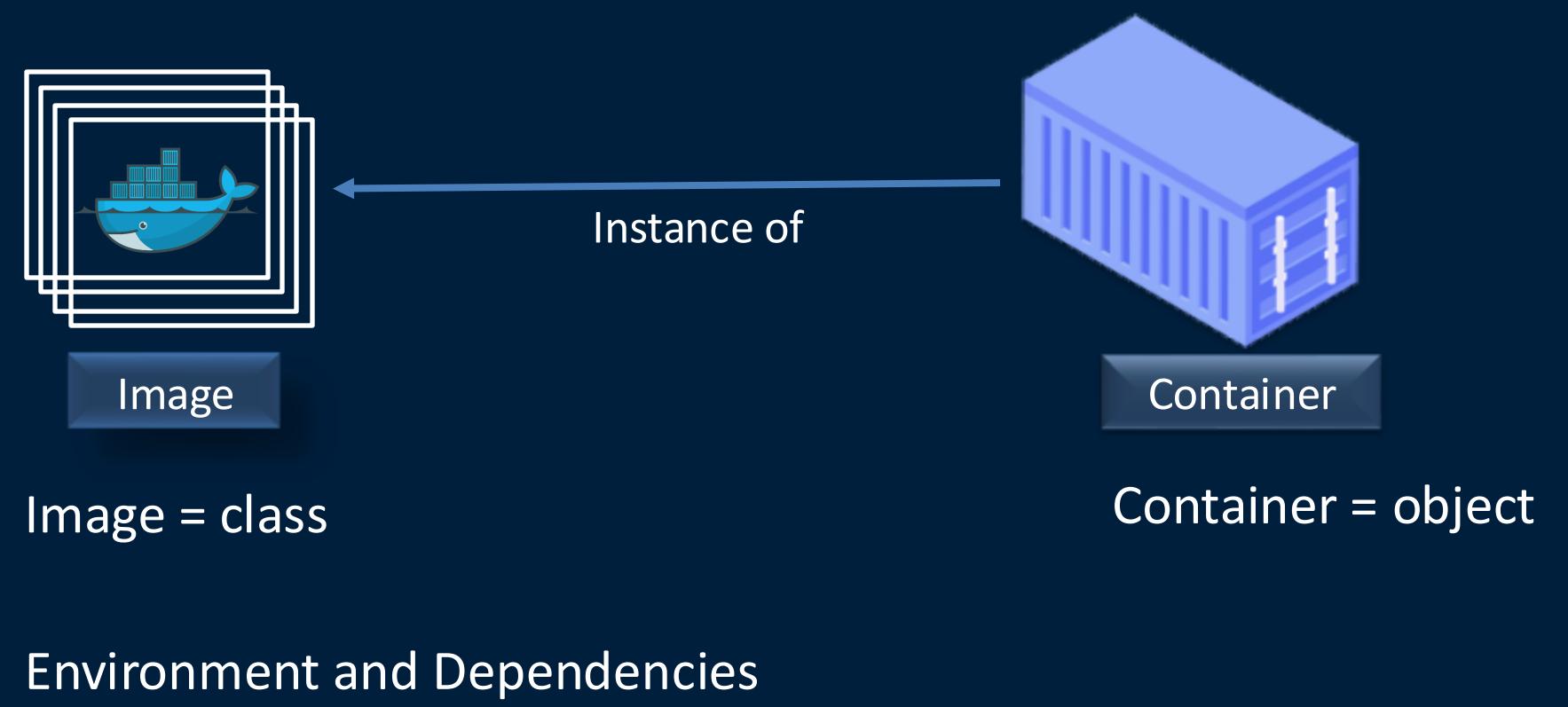


Overview of Docker Images

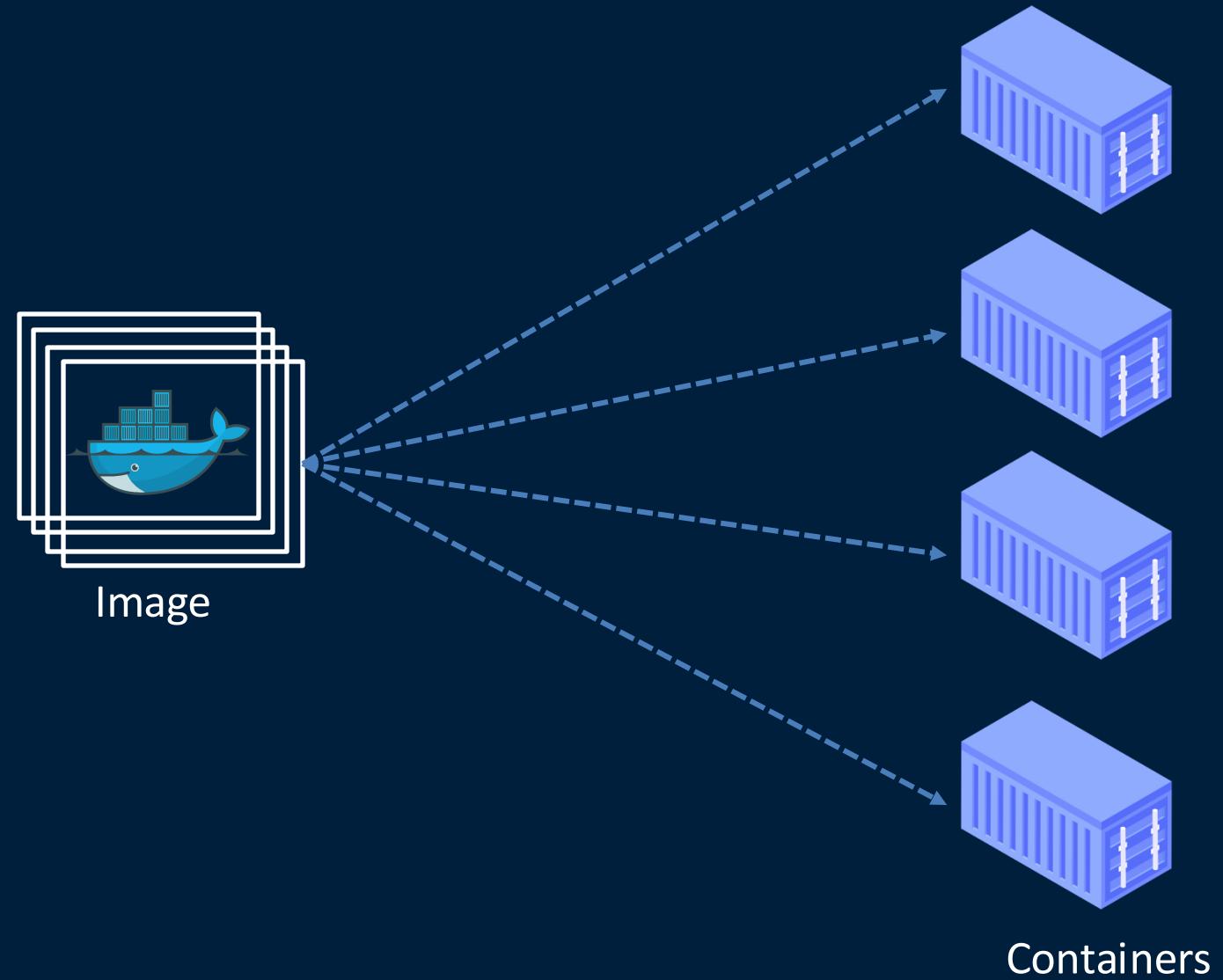


Docker images = Container images
Read-only templates

Images and Containers



Overview of Docker Images



Overview of Docker Images

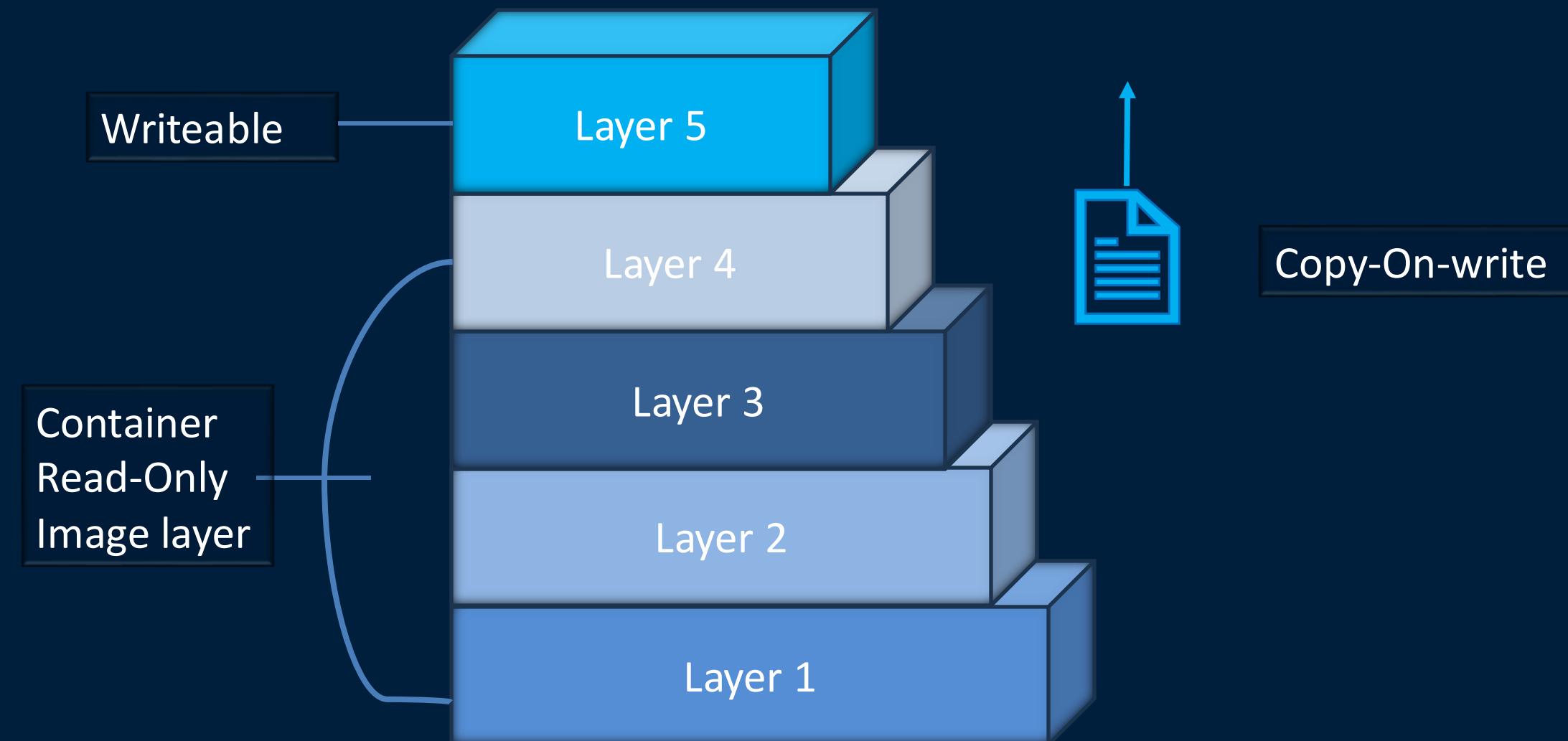


- Files
- Libraries
- Configurations

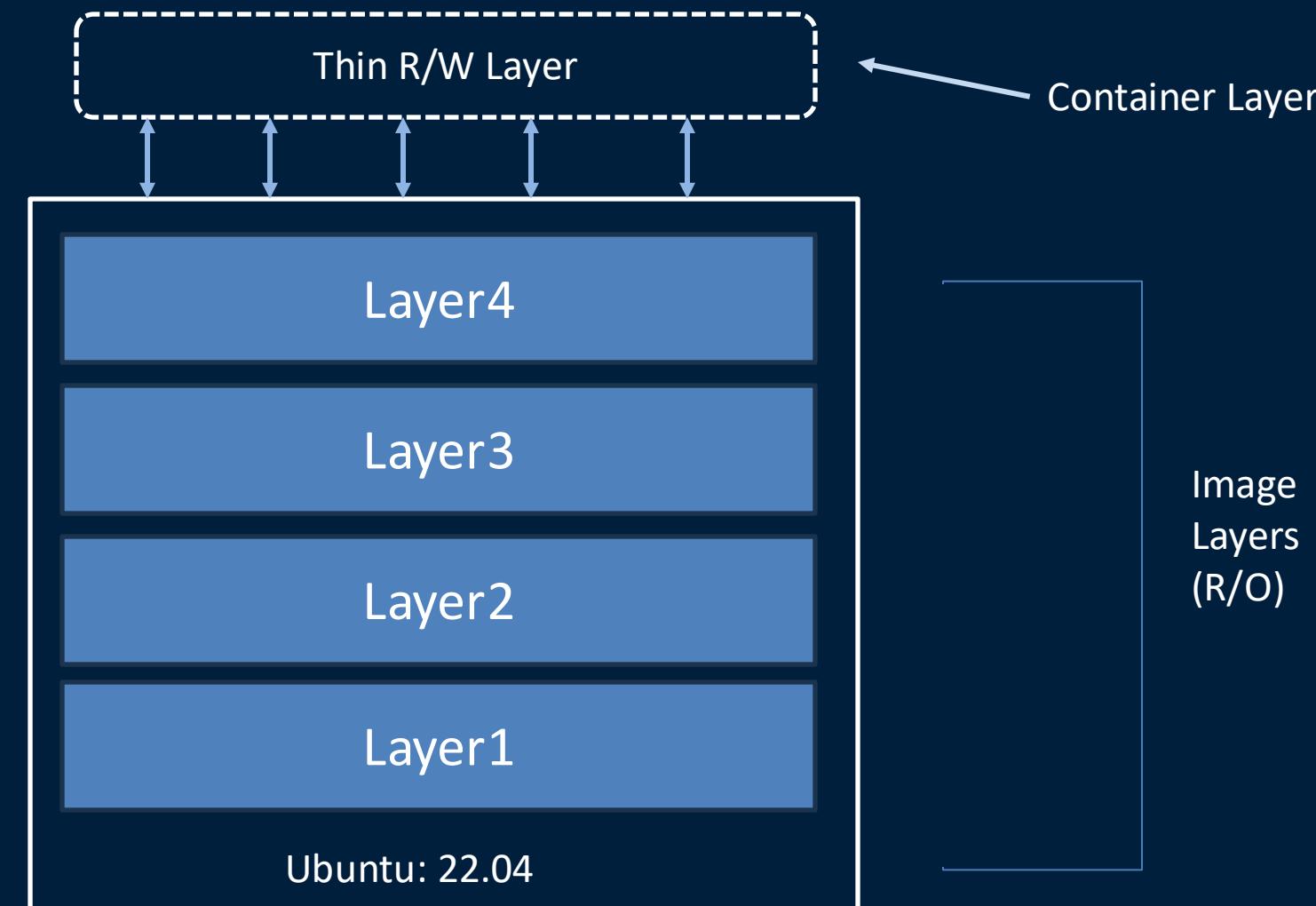


Overview of Docker Images

Union File System (OverlayFS)

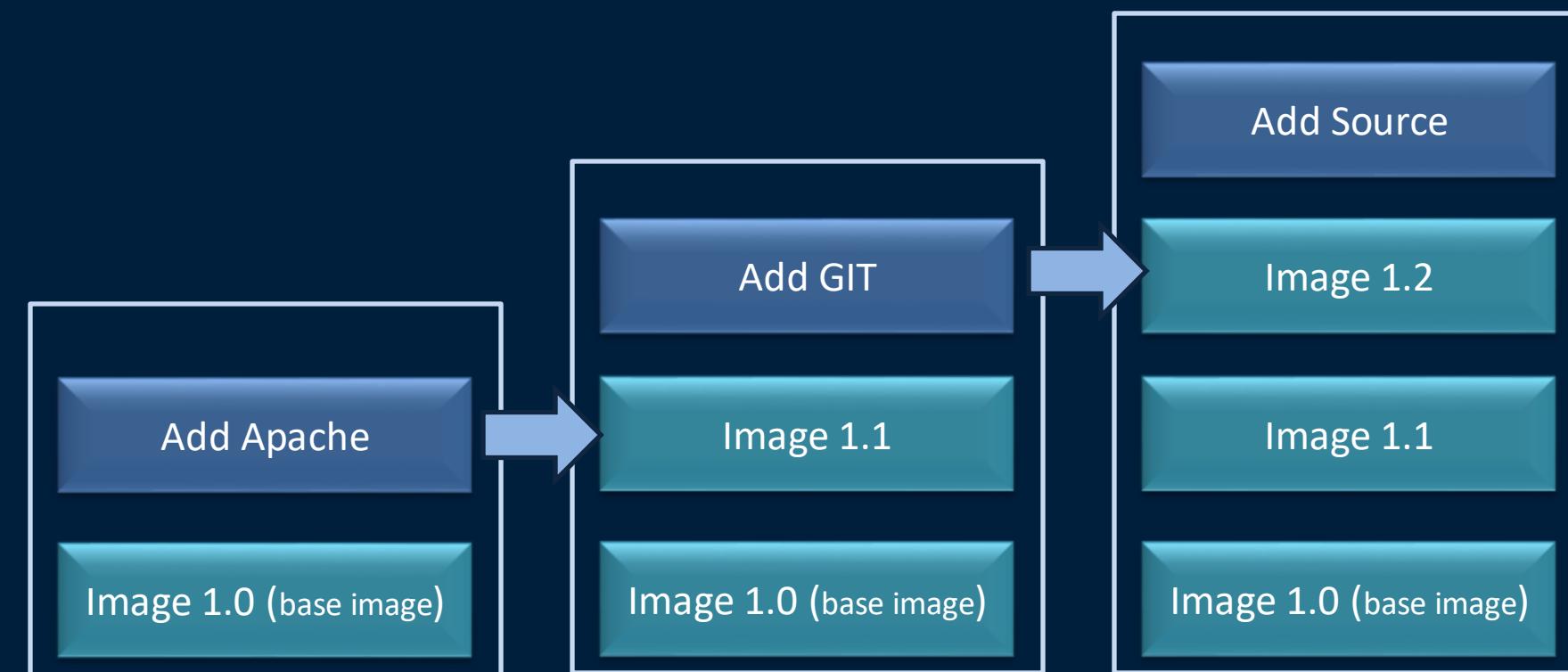


Overview of Docker Images



Overview of Docker Images

- ✓ Gradual modifications
- ✓ Reusability





Overview of Docker Images

Dockerfile

```
FROM centos:7

RUN yum -y update
RUN yum -y install httpd
RUN echo "Hello Docker" > /var/www/html/index.html

EXPOSE 8080

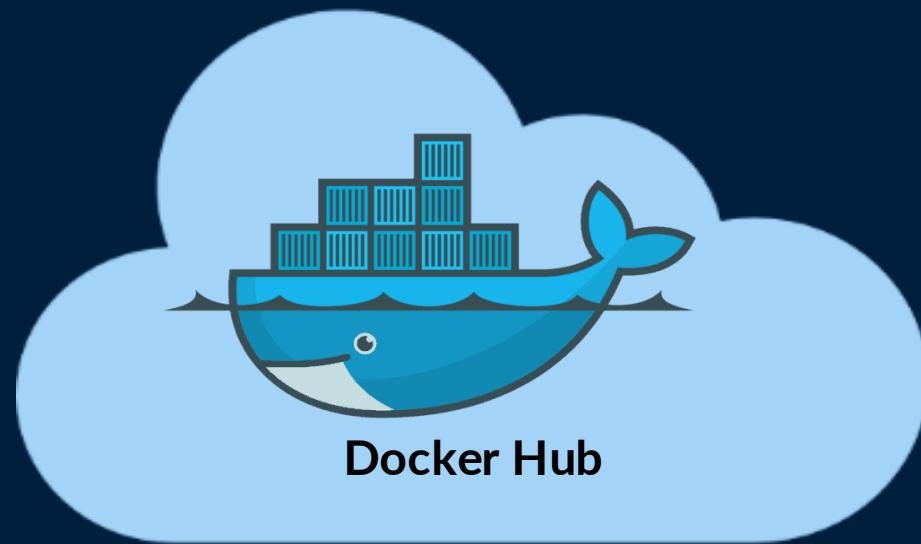
CMD ["httpd", "-D", "FOREGROUND"]
```

Overview of Docker Images

Registries

Dockerfile

```
FROM centos:7
RUN yum -y update
RUN yum -y install httpd
RUN echo "Hello Docker" > /var/www/html/index.html
EXPOSE 8080
CMD ["httpd","-D","FOREGROUND"]
```



Container Registry



Container Registry

- Store and share container images

Docker registry implementation



CLOUD NATIVE
COMPUTING FOUNDATION

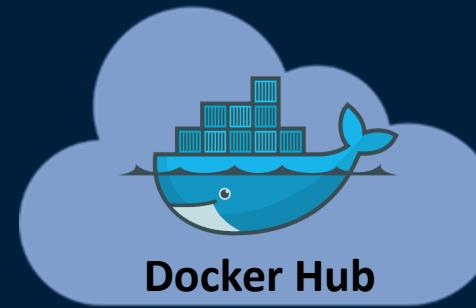
Distribution

Registry (Docker Hub)

Public

Private

Docker Registry



- Harbor
- JFrog Artifactory
- Quay.io
- GitHub Packages



Registry (Docker Hub)

Image Repository

Version Control

Integration

Collaboration

Web Interface

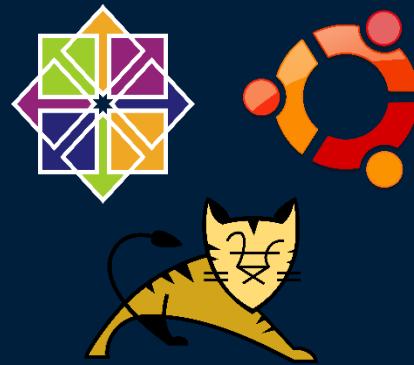
Automated Builds



Registry (Docker Hub)

Types of Images

Official



Verified
Publishers

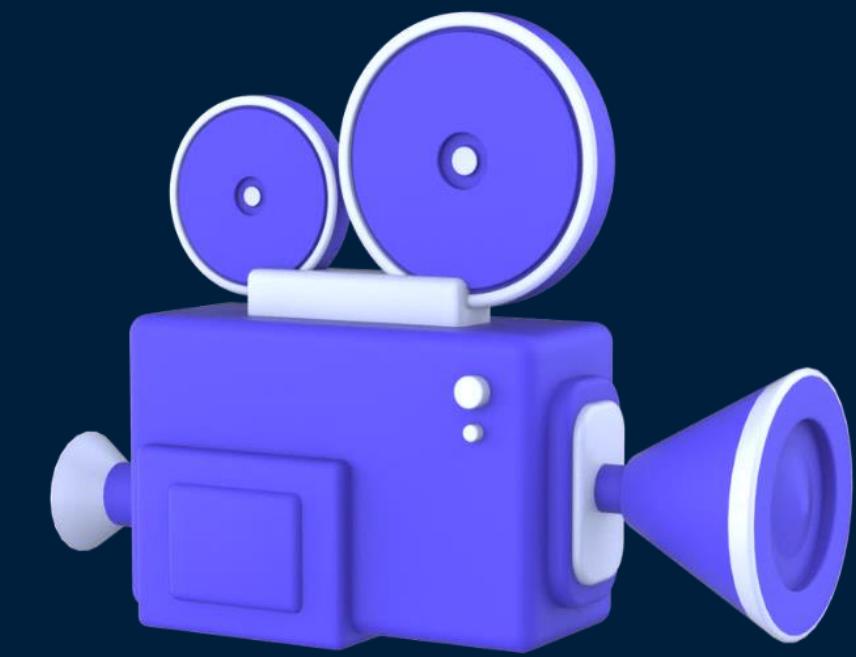


Sponsored
OSS

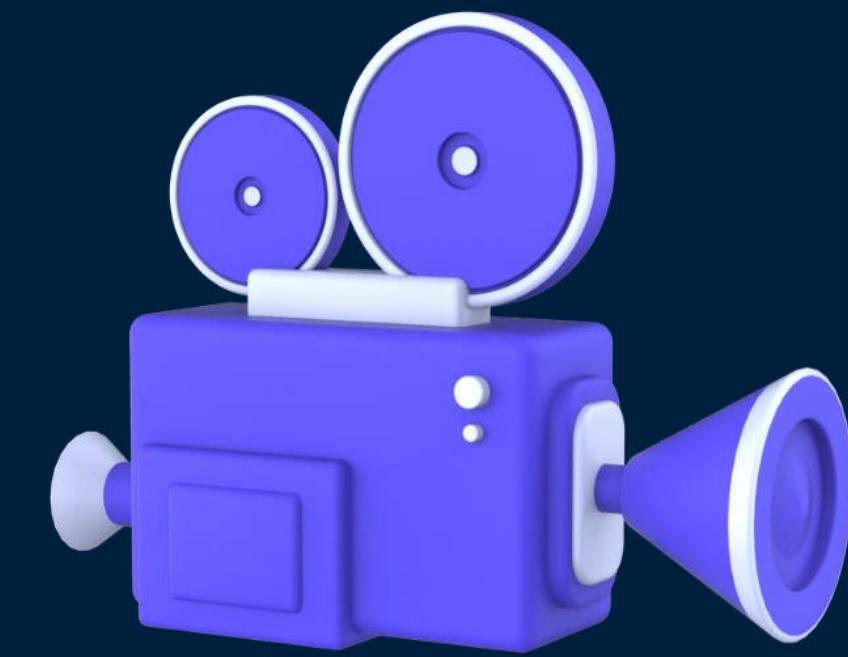


User

deepthianarayan/docker
thinknyx/jenkins



Demonstration | Docker Hub



Demonstration | Installing Docker



Managing images with Docker CLI

docker images <sub-command> [options]

docker images --help

```
Usage: docker image COMMAND

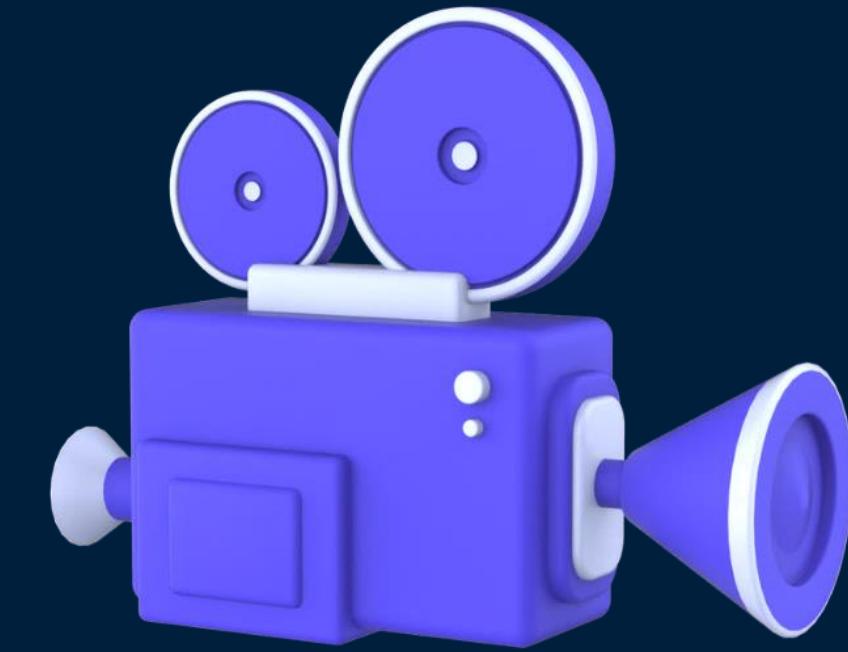
Manage images

Commands:
  build      Build an image from a Dockerfile
  history    Show the history of an image
  import     Import the contents from a tarball to create a filesystem image
  inspect    Display detailed information on one or more images
  load       Load an image from a tar archive or STDIN
  ls         List images
  prune     Remove unused images
  pull       Download an image from a registry
  push       Upload an image to a registry
  rm        Remove one or more images
  save      Save one or more images to a tar archive (streamed to STDOUT by default)
  tag       Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

Run 'docker image COMMAND --help' for more information on a command.
```

Managing images with Docker CLI

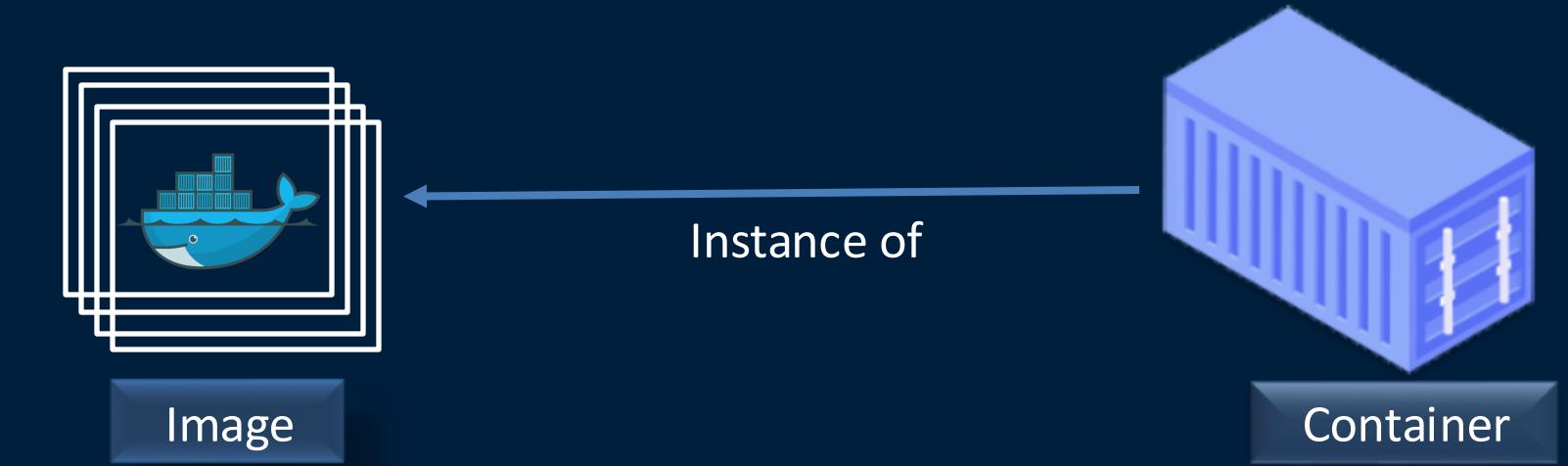
Description	Command	Alternative command
Download an image	<code>docker image pull <image_name></code>	<code>docker pull <image_name></code>
List all the images	<code>docker image ls</code>	<code>docker image list (or)</code> <code>docker images</code>
Tag or rename an image	<code>docker image tag <source_image> <target_image></code>	<code>docker tag <source_image> <target_image></code>
Show the history of an image	<code>docker image history <image_name></code>	<code>docker history <image_name></code>
Display a detailed information of an image	<code>docker image inspect <image_name></code>	<code>docker inspect <image_name></code>
Remove an image	<code>docker image rm <image_name></code>	<code>docker image remove <image_name></code> <code>(or) docker rmi <image_name></code>



Demonstration | Managing images with Docker CLI

Overview of Docker Containers

- Code
- Libraries
- Runtime
- Configurations



✓ Read-only templates

Overview of Docker Containers

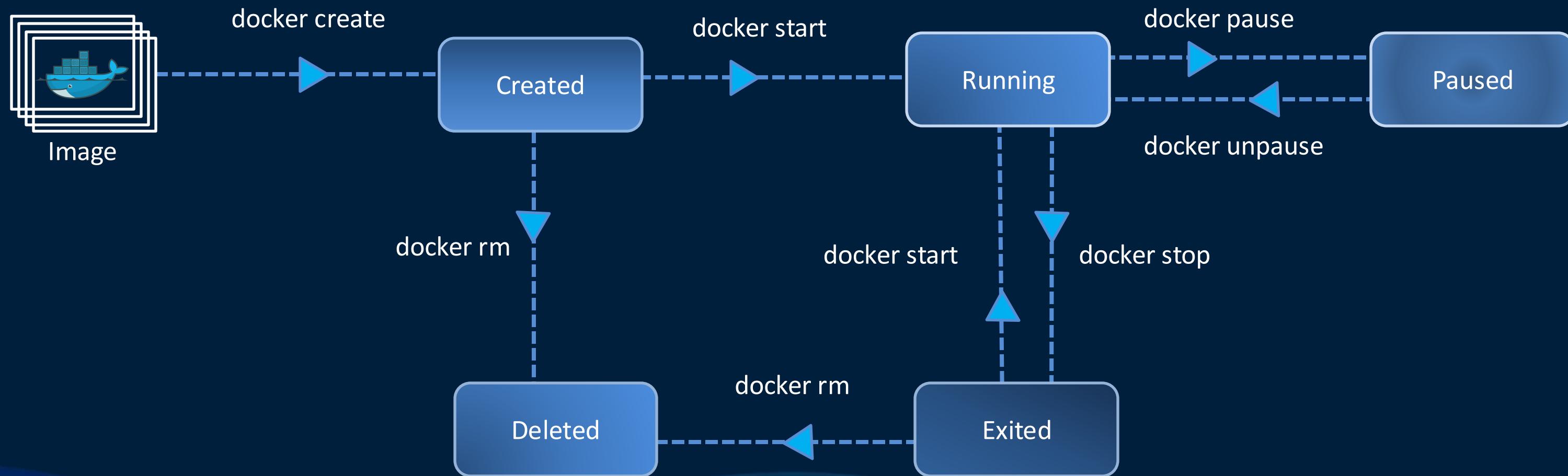
docker run <image_name>

docker run busybox



Overview of Docker Containers

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
1fa2a75ab62a	nginx	"/docker-entrypoint...."	11 seconds ago	Up 9 seconds	80/tcp	dreamy_edison



Managing containers with Docker CLI

docker container <sub-command> [options]

docker container --help

```
Usage: docker container COMMAND

Manage containers

Commands:
attach      Attach local standard input, output, and error streams to a running container
commit      Create a new image from a container's changes
cp          Copy files/folders between a container and the local filesystem
create      Create a new container
diff        Inspect changes to files or directories on a container's filesystem
exec        Execute a command in a running container
export      Export a container's filesystem as a tar archive
inspect    Display detailed information on one or more containers
kill        Kill one or more running containers
logs        Fetch the logs of a container
ls          List containers
pause      Pause all processes within one or more containers
port        List port mappings or a specific mapping for the container
prune      Remove all stopped containers
rename     Rename a container
restart    Restart one or more containers
rm         Remove one or more containers
run         Create and run a new container from an image
start      Start one or more stopped containers
stats      Display a live stream of container(s) resource usage statistics
stop       Stop one or more running containers
top        Display the running processes of a container
unpause   Unpause all processes within one or more containers
update     Update configuration of one or more containers
wait       Block until one or more containers stop, then print their exit codes

Run 'docker container COMMAND --help' for more information on a command.
```

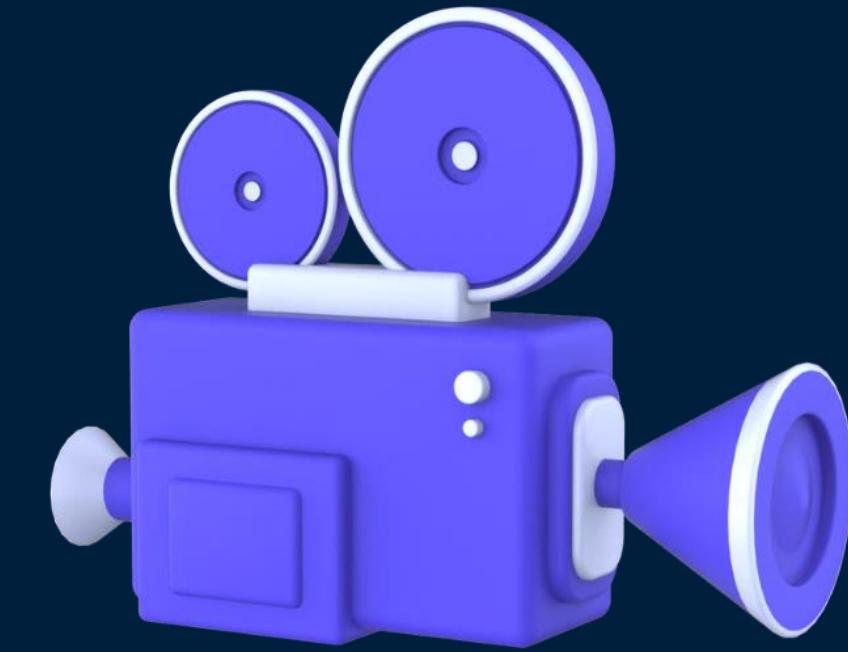
docker container run
↑
↓
docker run

Managing containers with Docker CLI

Description	Command	Alternative command
Create and start a container	<code>docker container create <image_name></code> <code>docker container start <container_name></code>	<code>docker create <image_name></code> <code>docker start <container_name></code>
Run a container	<code>docker container run <image_name></code>	<code>docker run <image_name></code>
List all the containers	<code>docker container ls -a</code>	<code>docker container list -a</code> <code>docker container ps -a</code> <code>docker ps -a</code>
Stop a container	<code>docker container stop <container_name></code>	<code>docker stop <container_name></code>
Execute a command in a running container	<code>docker container exec <container_name> [ARG]</code>	<code>docker exec <container_name> [ARG]</code>
Rename a container	<code>docker container rename <old_container_name> <new_container_name></code>	<code>docker rename <old_container_name> <new_container_name></code>

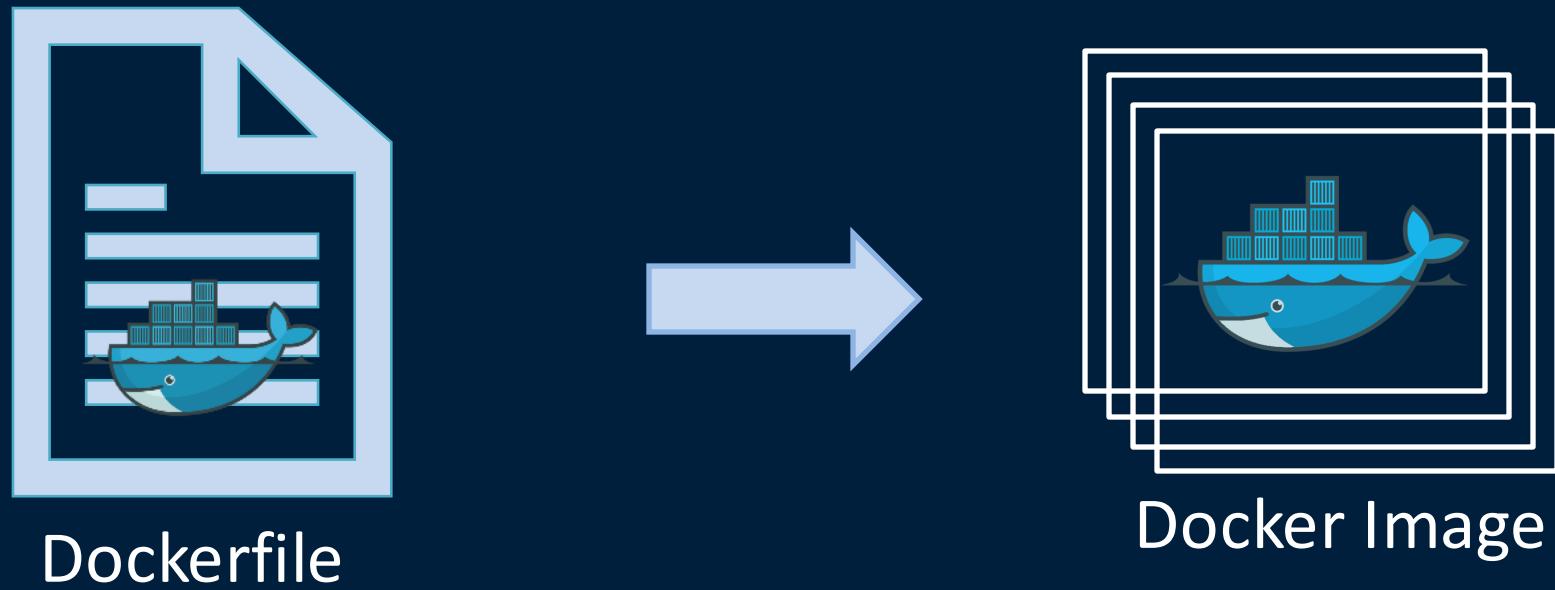
Managing containers with Docker CLI

Description	Command	Alternative command
View the logs of a container	<code>docker container logs <container_name></code>	<code>docker logs <container_name></code>
View the detailed information about a container	<code>docker container inspect <container_name></code>	<code>docker inspect <container_name></code>
Remove a Container	<code>docker container rm <container_name></code>	<code>docker container remove <container_name></code> <code>docker rm <container_name></code>



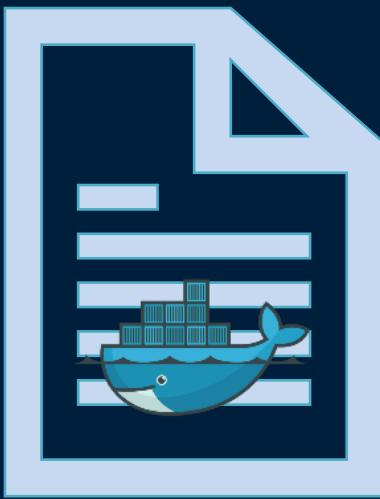
Demonstration | Managing containers with Docker CLI

Getting Started with Dockerfile



- Script with instructions to construct the image

Getting Started with Dockerfile



Dockerfile

Build process

Images are consistent, reproducible, and ready to deploy across any environment

Steps to construct Docker image

- Setting up environment
- Installing dependencies
- Copying application files

Getting Started with Dockerfile

INSTRUCTION arguments

```
FROM busybox
```



Dockerfile

Instruction	Description
ADD	Add local or remote files and directories.
ARG	Use build-time variables.
CMD	Specify default commands.
COPY	Copy files and directories.
ENTRYPOINT	Specify default executable.
ENV	Set environment variables.
EXPOSE	Describe which ports your application is listening on.
FROM	Create a new build stage from a base image.
HEALTHCHECK	Check a container's health on startup.
LABEL	Add metadata to an image.
MAINTAINER	Specify the author of an image.
ONBUILD	Specify instructions for when the image is used in a build.
RUN	Execute build commands.
SHELL	Set the default shell of an image.
STOP SIGNAL	Specify the system call signal for exiting a container.
USER	Set user and group ID.
VOLUME	Create volume mounts.
WORKDIR	Change working directory.

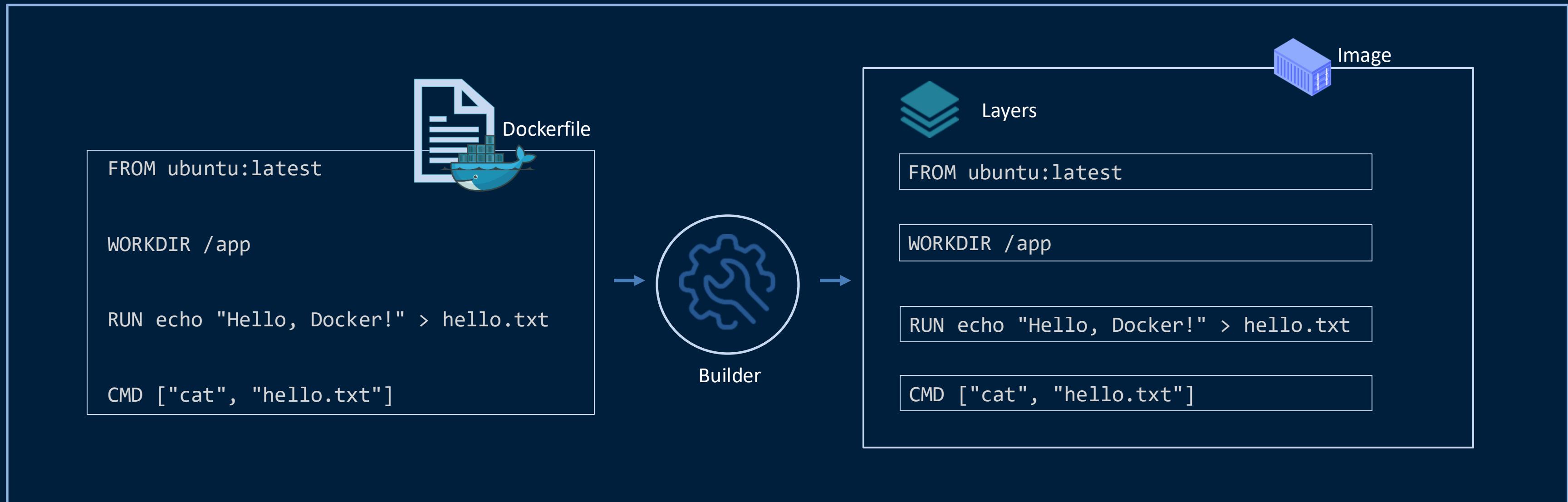


Getting Started with Dockerfile

```
FROM ubuntu:latest  
  
WORKDIR /app  
  
RUN echo "Hello, Docker!" > hello.txt  
  
CMD ["cat", "hello.txt"]
```

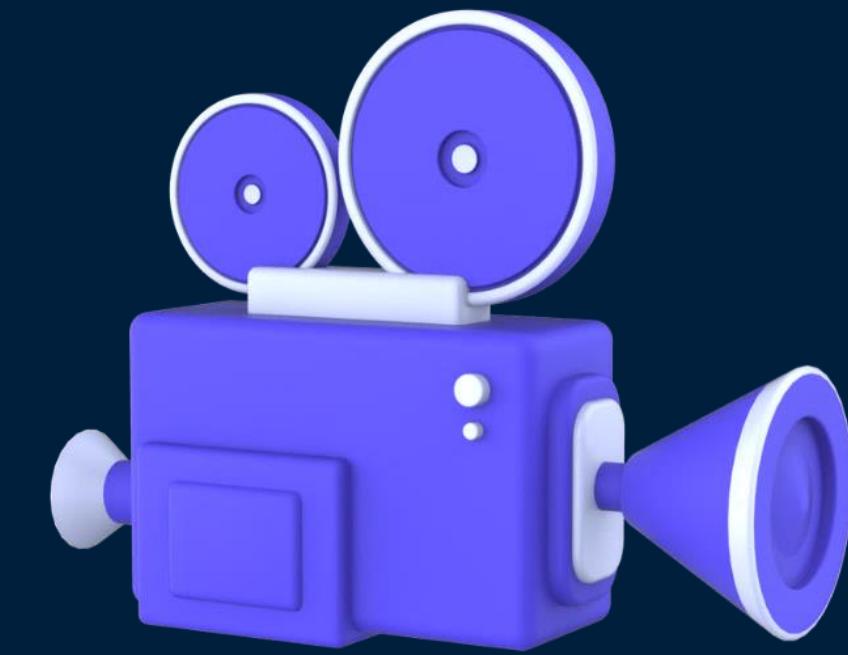
- Dockerfile instructions are executed sequentially
- Each instruction translates to an image layer

Getting Started with Dockerfile

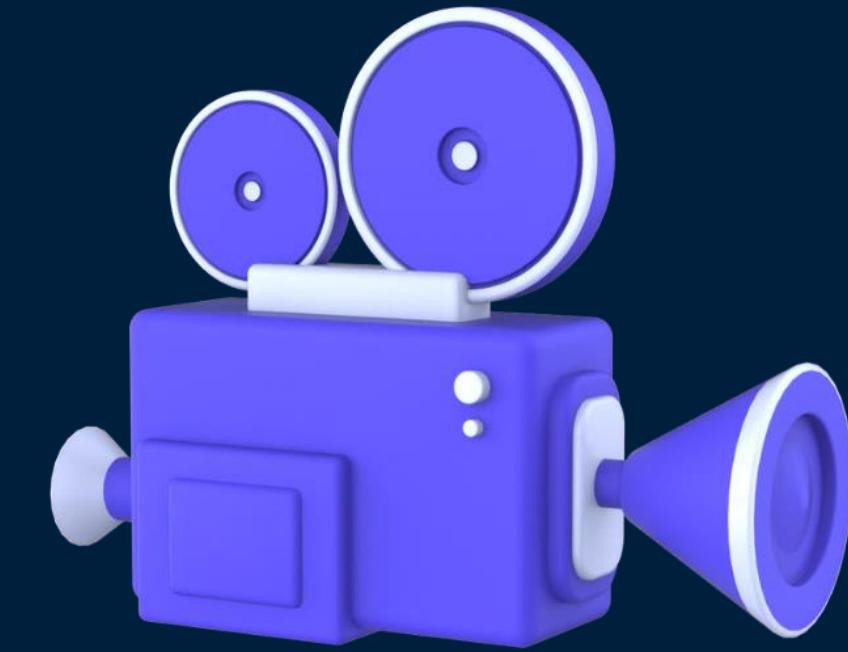


Getting Started with Dockerfile

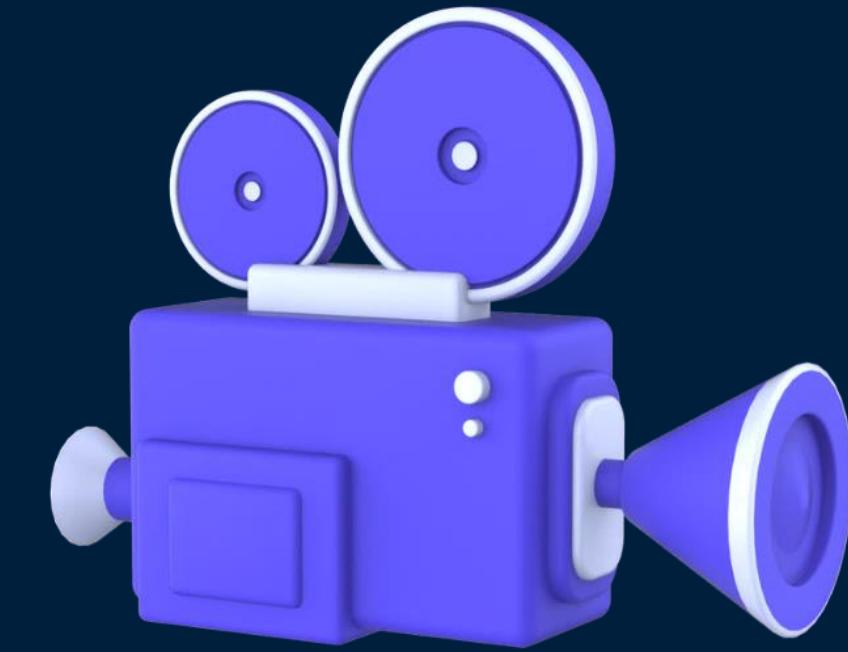
Description	Command	Alternative command
Build an image from a Dockerfile	<code>docker image build -t <image_name> .</code>	<code>docker build (or)</code> <code>docker buildx build (or)</code> <code>docker builder build</code>



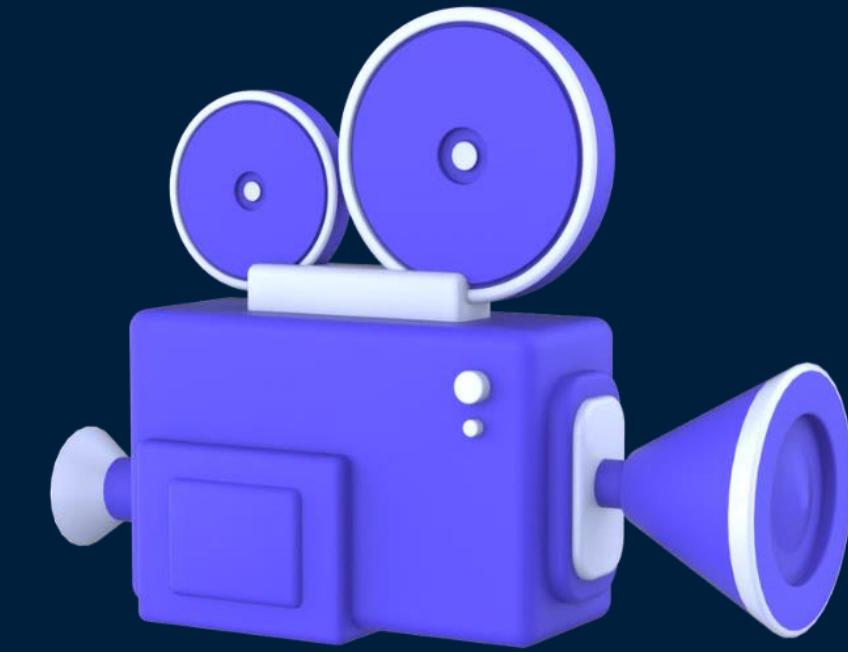
Demonstration | Creating a Dockerfile



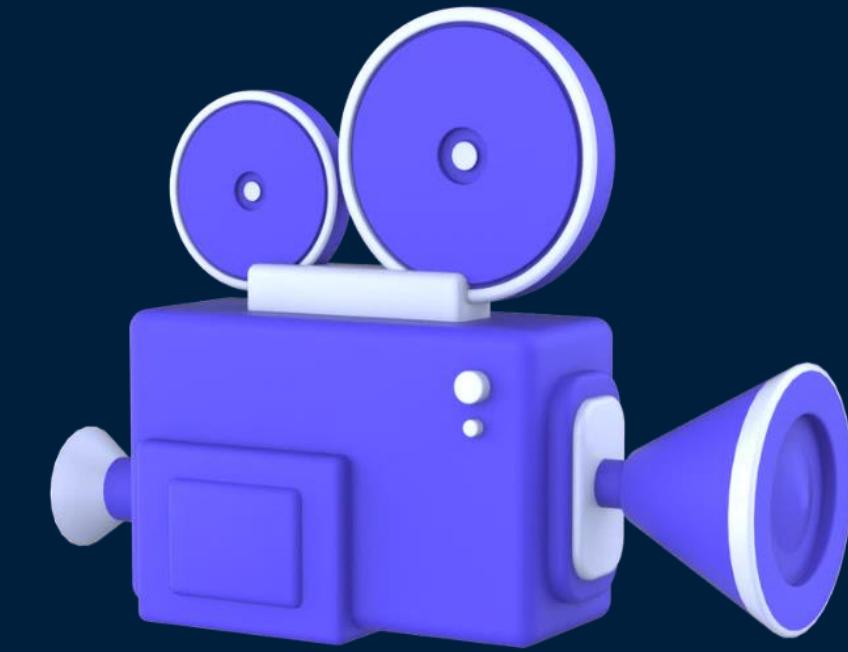
Demonstration | Validating Dockerfile and building image



Demonstration | Running a Container from our image



Demonstration | Publishing Image to a Registry



Demonstration | Containerize the Python Flask app with Docker



Summary

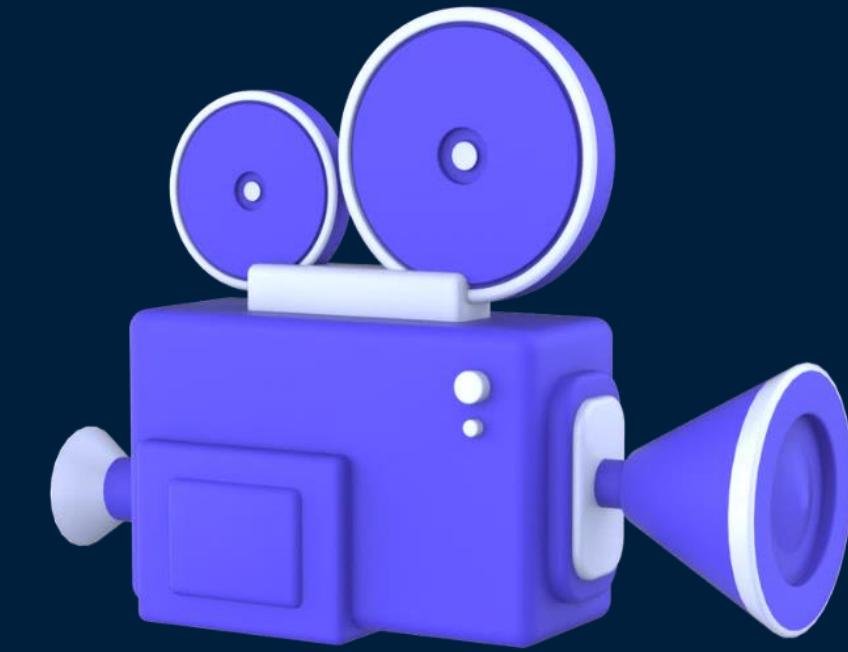
- ✓ Fundamentals of containerization and importance of Docker in DevOps
- ✓ Key Docker concepts, architecture, and objects
- ✓ Docker images, containers, and registries
- ✓ Installing Docker and managing images/containers via CLI
- ✓ Creating and validating Dockerfiles
- ✓ Building images and running containers
- ✓ Publishing images to Docker Hub



Containerizing a Python Flask application



Mastering Docker Essentials – Hands-on DevOps



Demonstration | Updating Docker Progress in GitHub Projects



Thank You

Kubernetes for Orchestration

By Thinknyx Technologies LLP

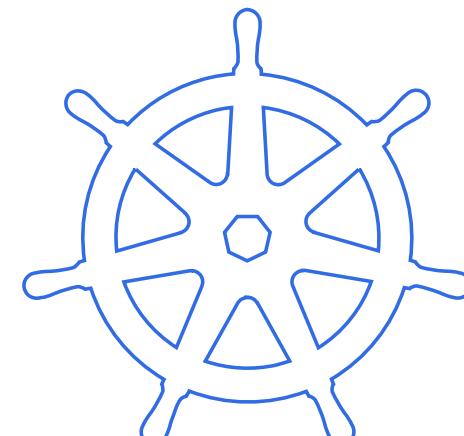
Section 8



Getting Started with Kubernetes

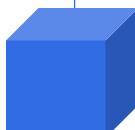
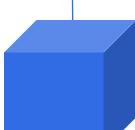
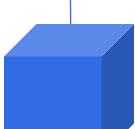
Section Overview

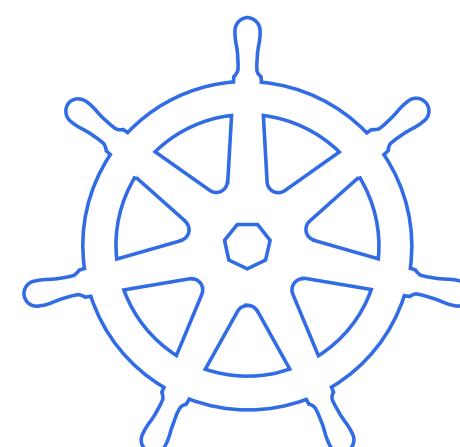
- † **Leading container orchestration platform**
- † **Manages and scales containerized applications reliably**
- † **Architecture and key objects**
- † **Hands-on practice with cluster setup**
- † **Working with namespaces and pods**
- † **Managing ReplicaSets and deployments**
- † **Using labels and selectors**
- † **Understanding services**
- † **Developing Kubernetes Object manifest files in YAML**



Limitations of Containers

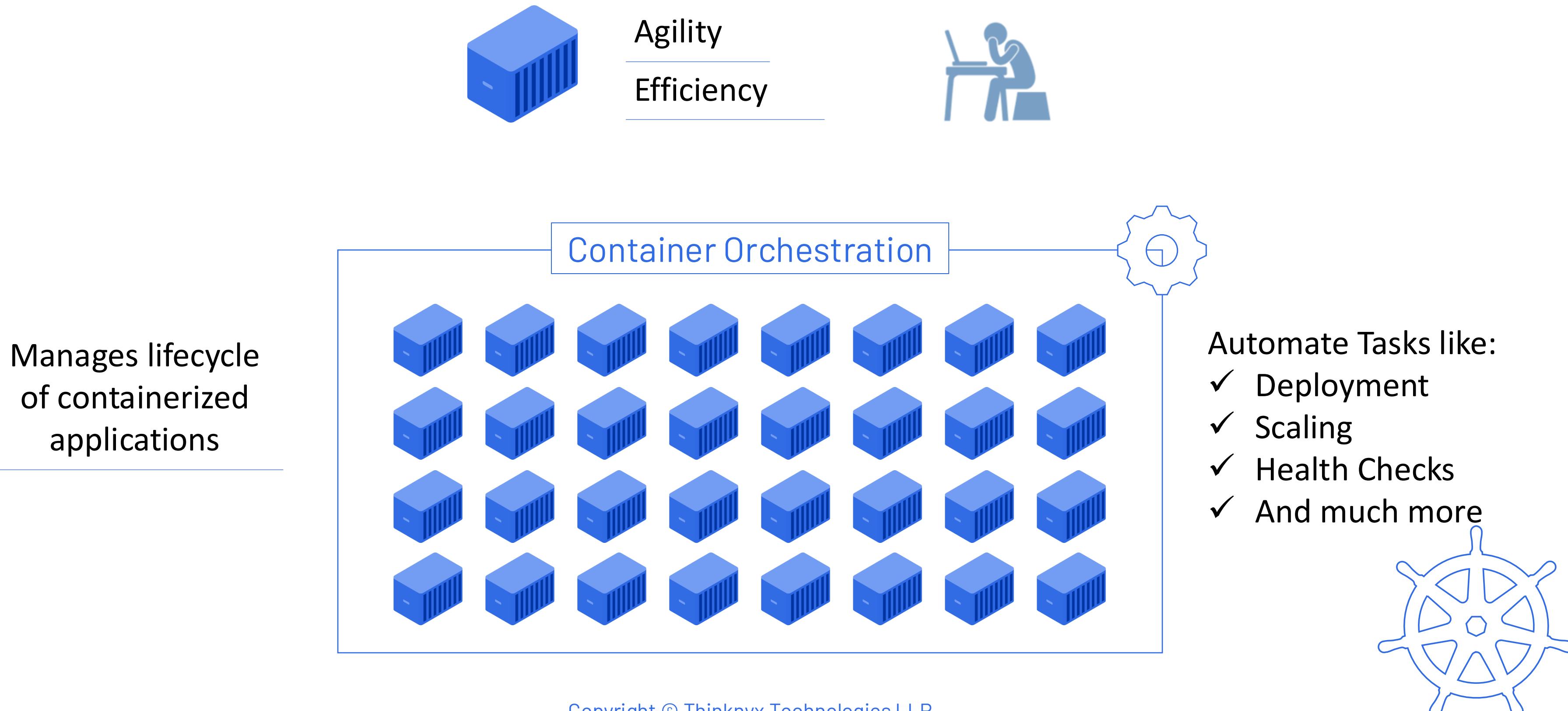
Limitations of Containers

-  **Self Healing**
 - No built-in mechanism to automatically re-create containers
-  **High Availability**
 - When server fails, entire application becomes unavailable, causing significant downtime
-  **Scalability**
 - Lacks automatic scaling
-  **Load Balancing**
 - Doesn't directly handle load balancing,
-  **Storage Constraints**
 - Rely on the underlying host system's storage resources



Container Orchestration

Container Orchestration

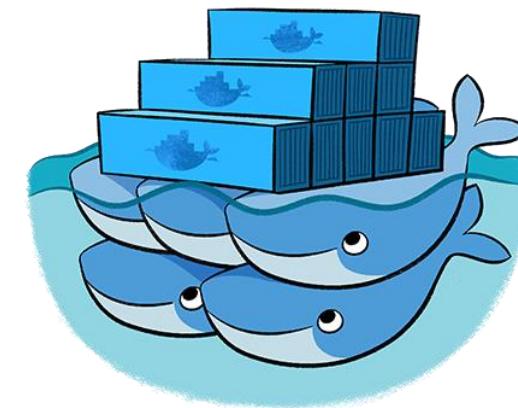


Container Orchestration



Kubernetes

- ✓ The Orchestral Maestro
- ✓ High availability, automated scaling, self-healing capabilities, and rich service discovery mechanisms
- ✓ Steeper learning curve



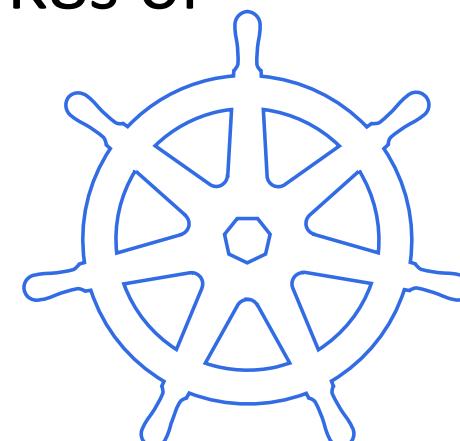
Docker Swarm

- ✓ Native container orchestration platform
- ✓ Easy to set up and use
- ✓ Less extensive compared to K8s



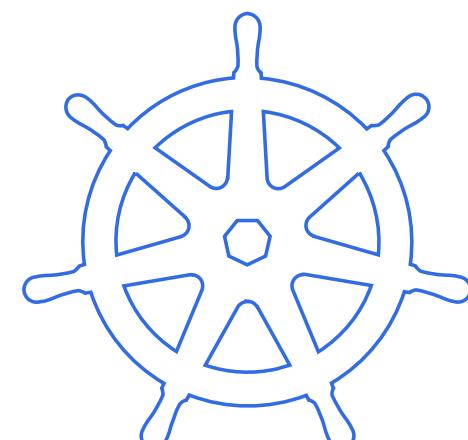
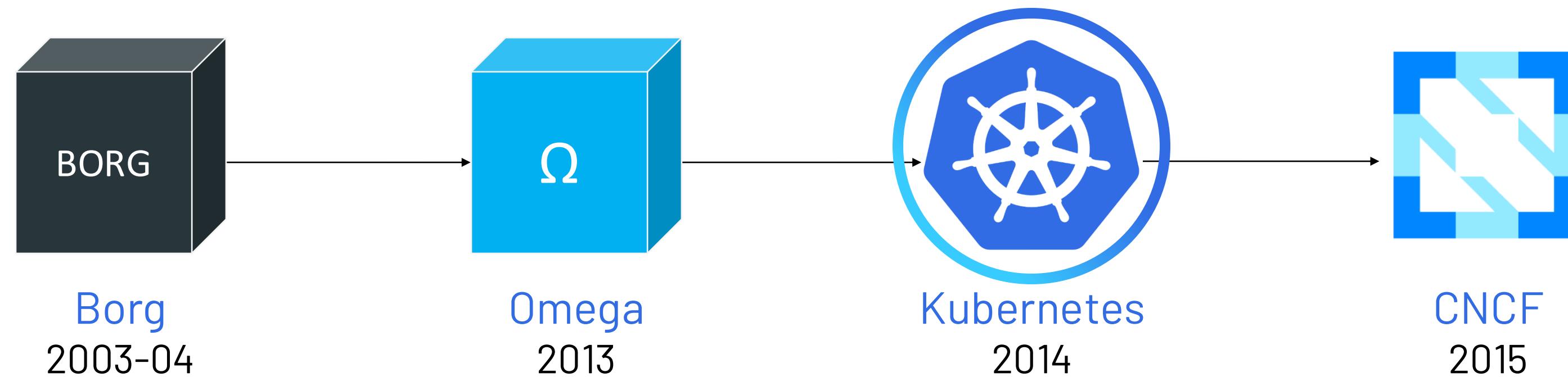
Apache Mesos

- ✓ Acts as a cluster kernel
- ✓ Flexible and resource optimization
- ✓ Requires more configuration effort compared to K8s or Swarm



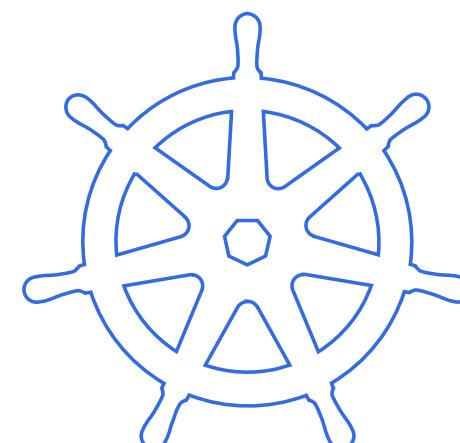
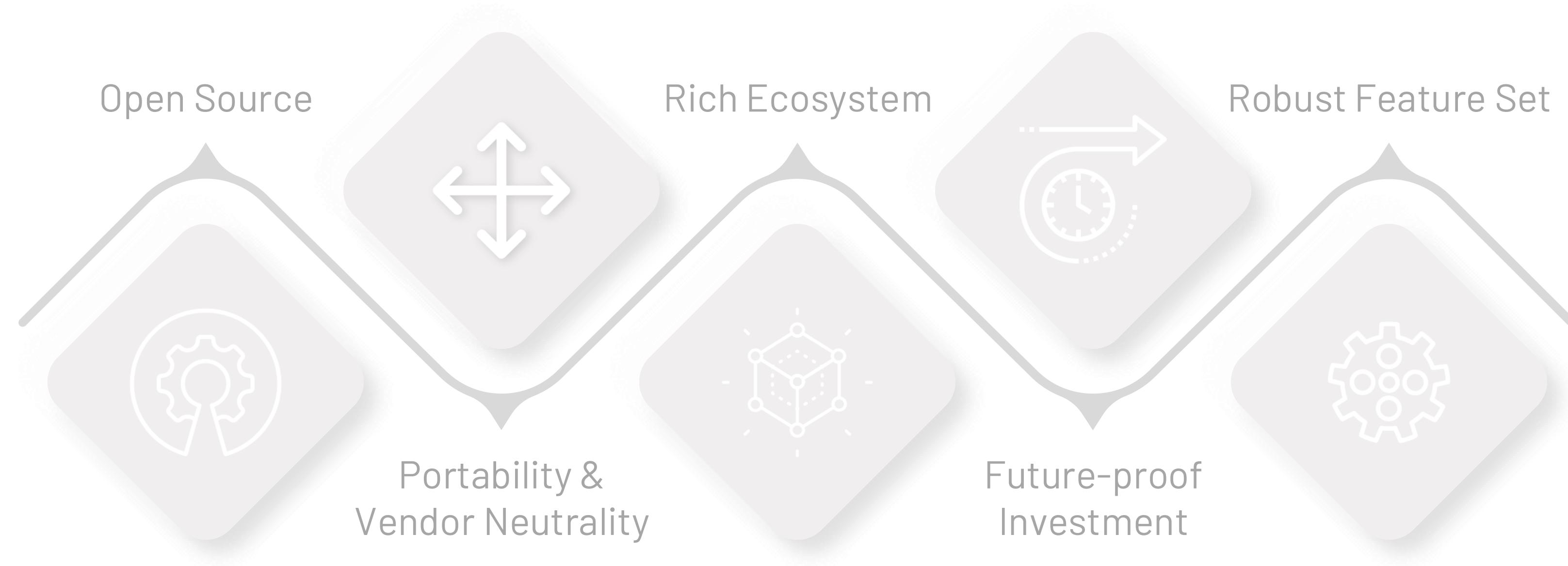
Introduction to Kubernetes

Introduction to Kubernetes



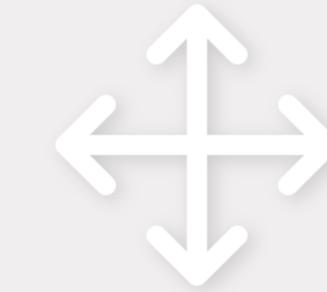
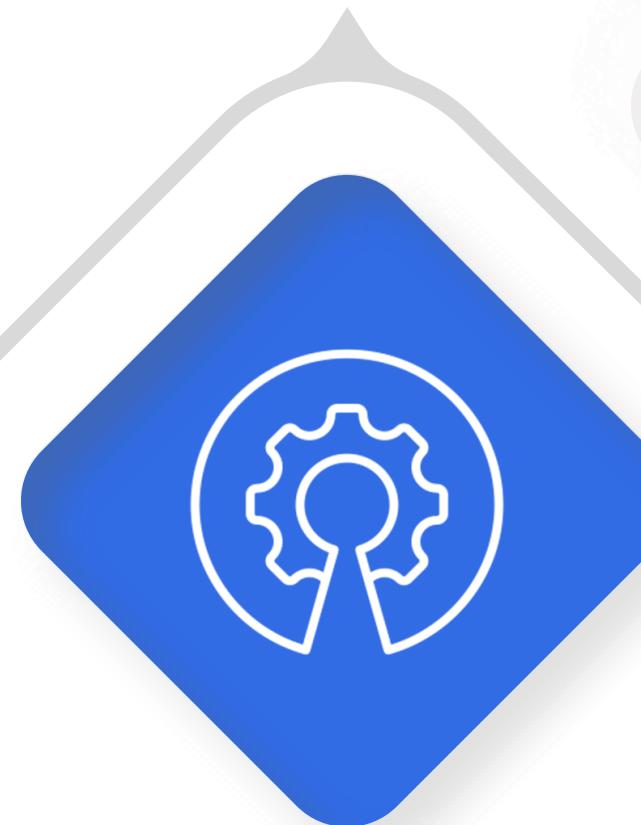
Why Kubernetes?

Why Kubernetes?



Why Kubernetes?

Open Source



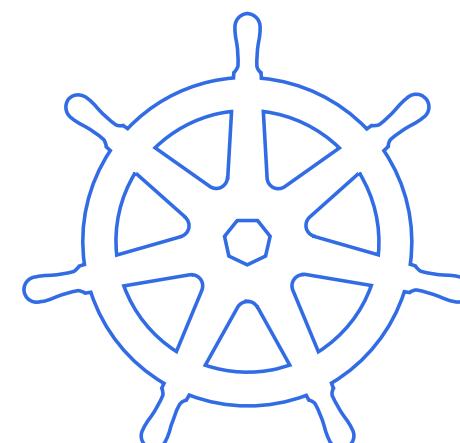
Portability &
Vendor Neutrality

Rich Ecosystem

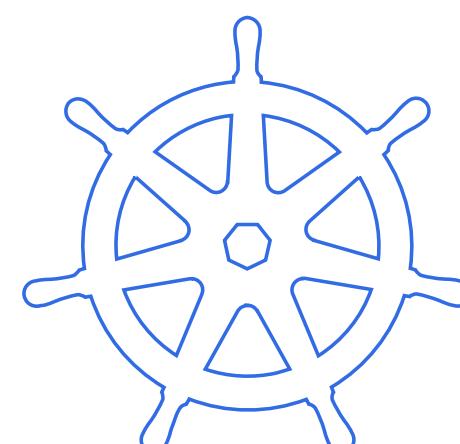


Future-proof
Investment

Robust Feature Set

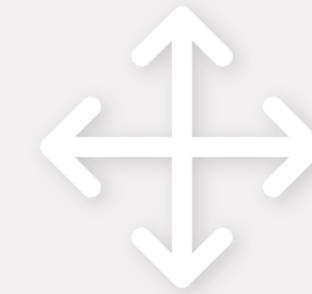


Why Kubernetes?



Why Kubernetes?

Open Source



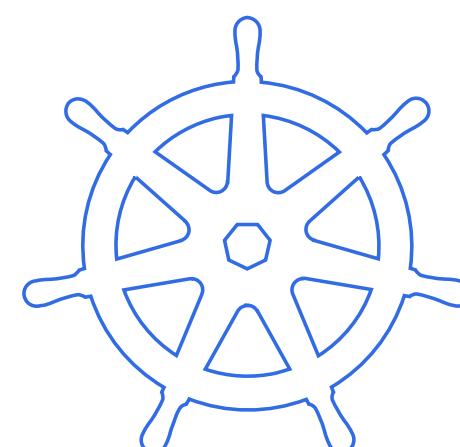
Portability &
Vendor Neutrality

Rich Ecosystem

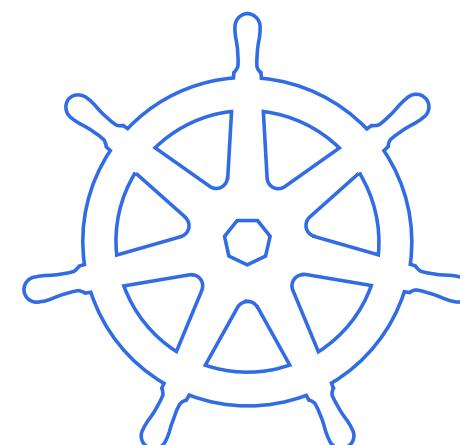


Future-proof
Investment

Robust Feature Set

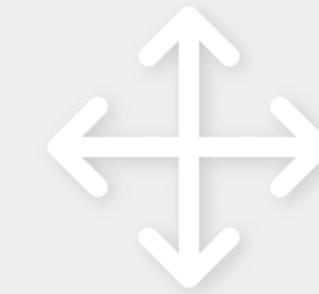


Why Kubernetes?



Why Kubernetes?

Open Source



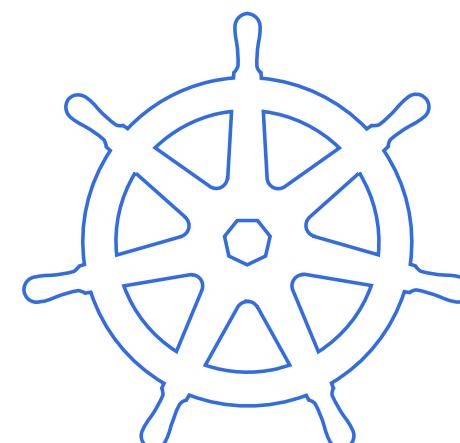
Portability &
Vendor Neutrality

Rich Ecosystem



Future-proof
Investment

Robust Feature Set



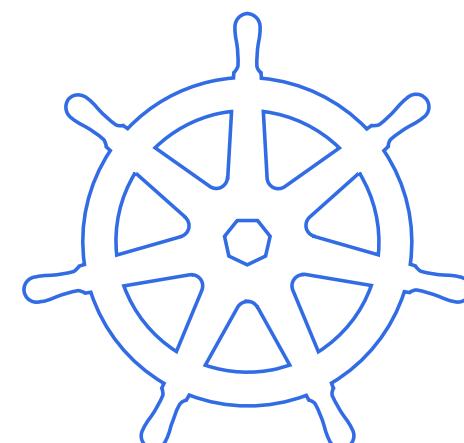
Setup Options

Setup Options

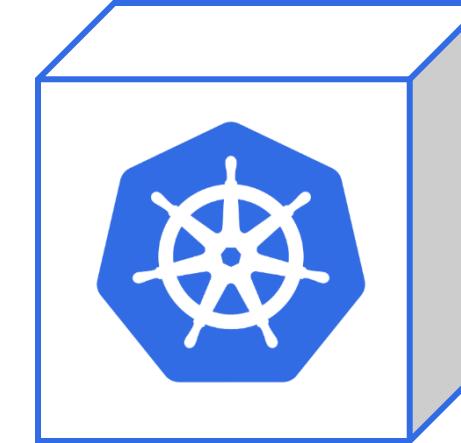


Native Kubernetes

- ✓ Full Control & Flexibility at no cost
- ✓ Requires Hands-On exposure
- ✓ Ideal for experienced users
- ✓ No official support (Only Community driven support)

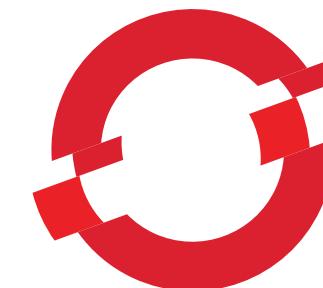


Setup Options

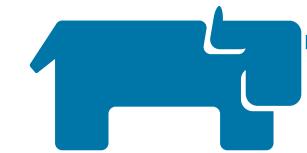


Enterprise Kubernetes

- ✓ Built upon Native Kubernetes
- ✓ Comes with official support
- ✓ Higher costs
- ✓ Higher resource requirements (for control plane and worker nodes)
- ✓ Ideal for Enterprises requiring immediate support and willing to make substantial investments



OpenShift



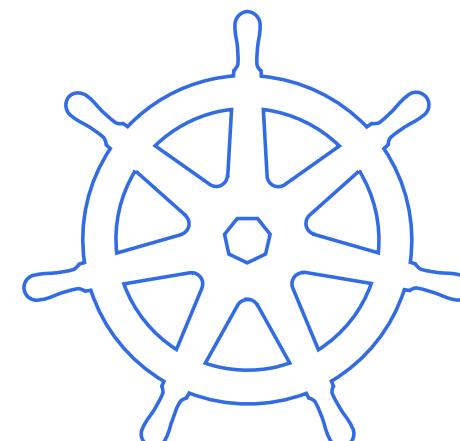
Rancher



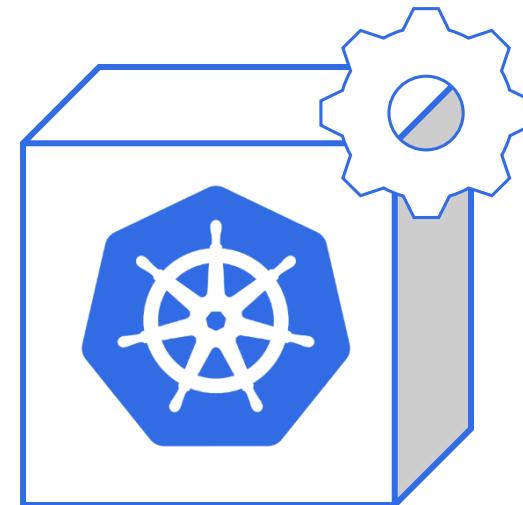
Tanzu



Rakuten CNP



Setup Options

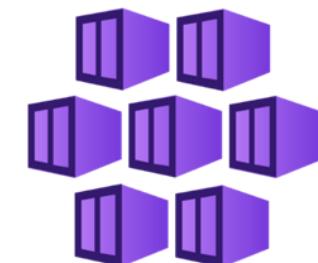


Managed Kubernetes

- ✓ Popularly known as Container as a service(CaaS)
- ✓ Focus only on Applications development & deployment
- ✓ Integrating toolsets may require additional efforts
- ✓ Cost-effective for smaller deployments
- ✓ Ideal for organizations looking for a straightforward entry into Kubernetes



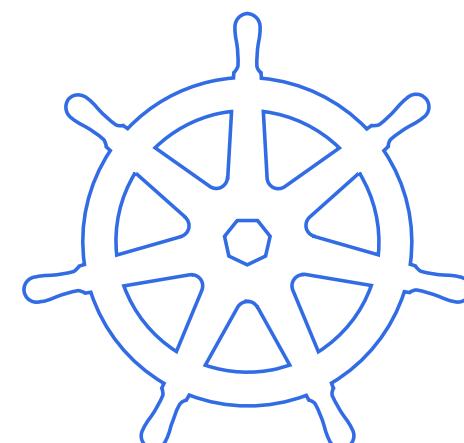
EKS



AKS

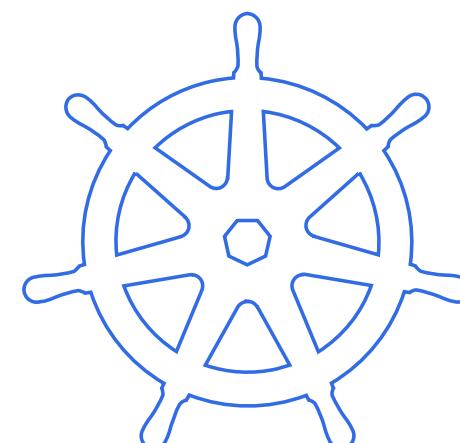
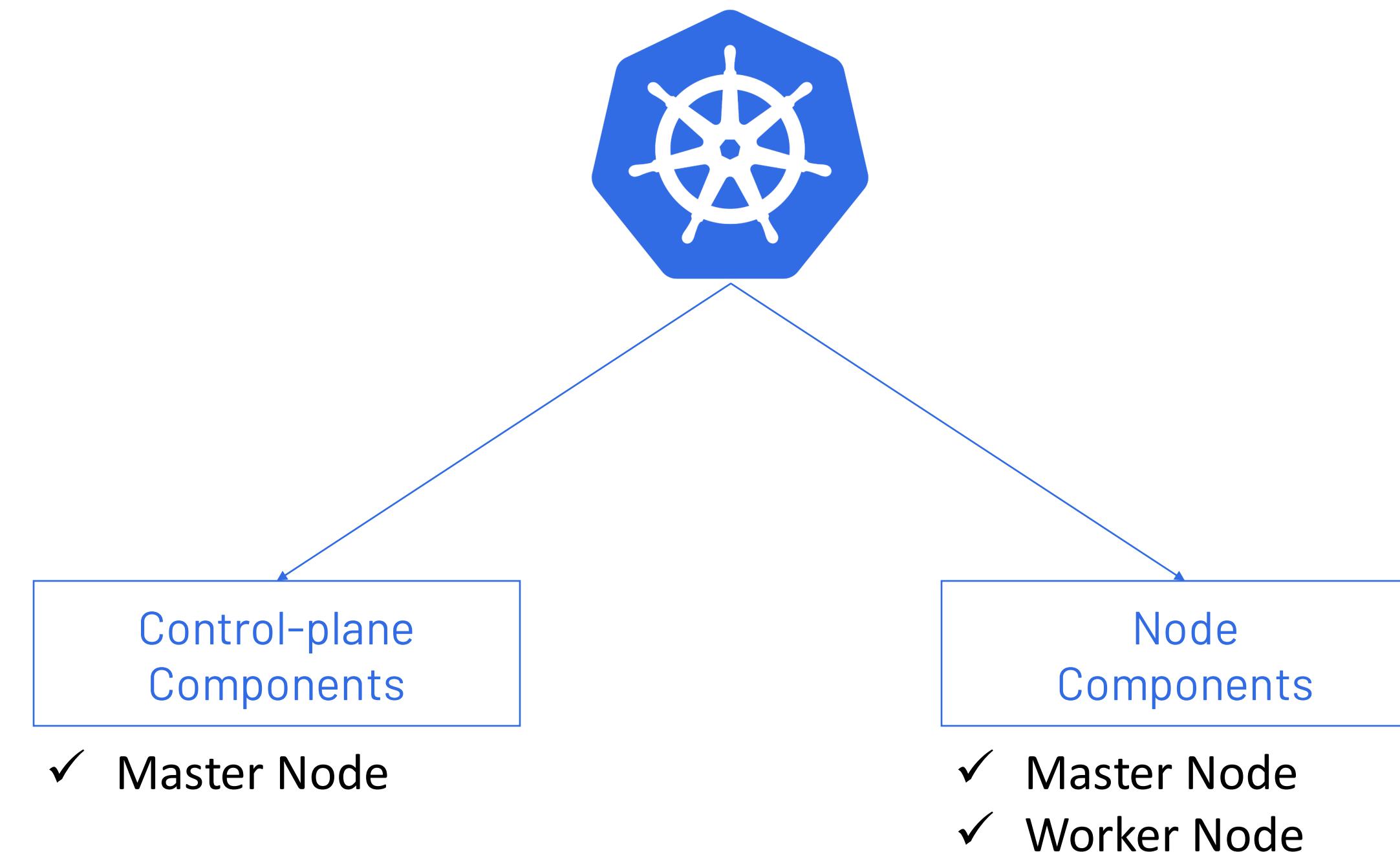


GKE



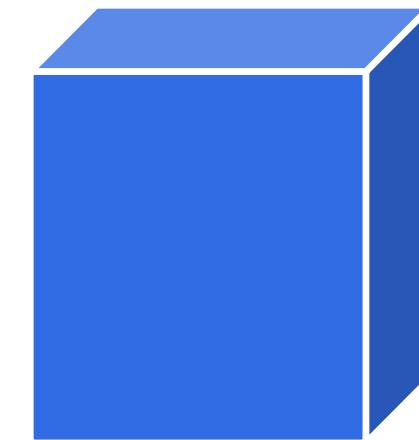
Overview of Kubernetes Components

Overview of Kubernetes Components



Control-plane Components

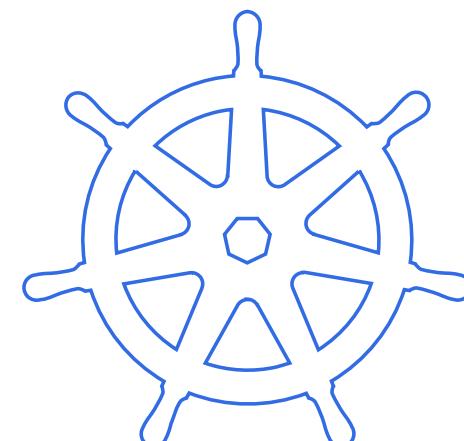
Control-plane Components



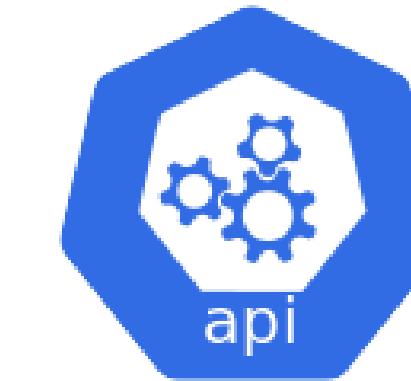
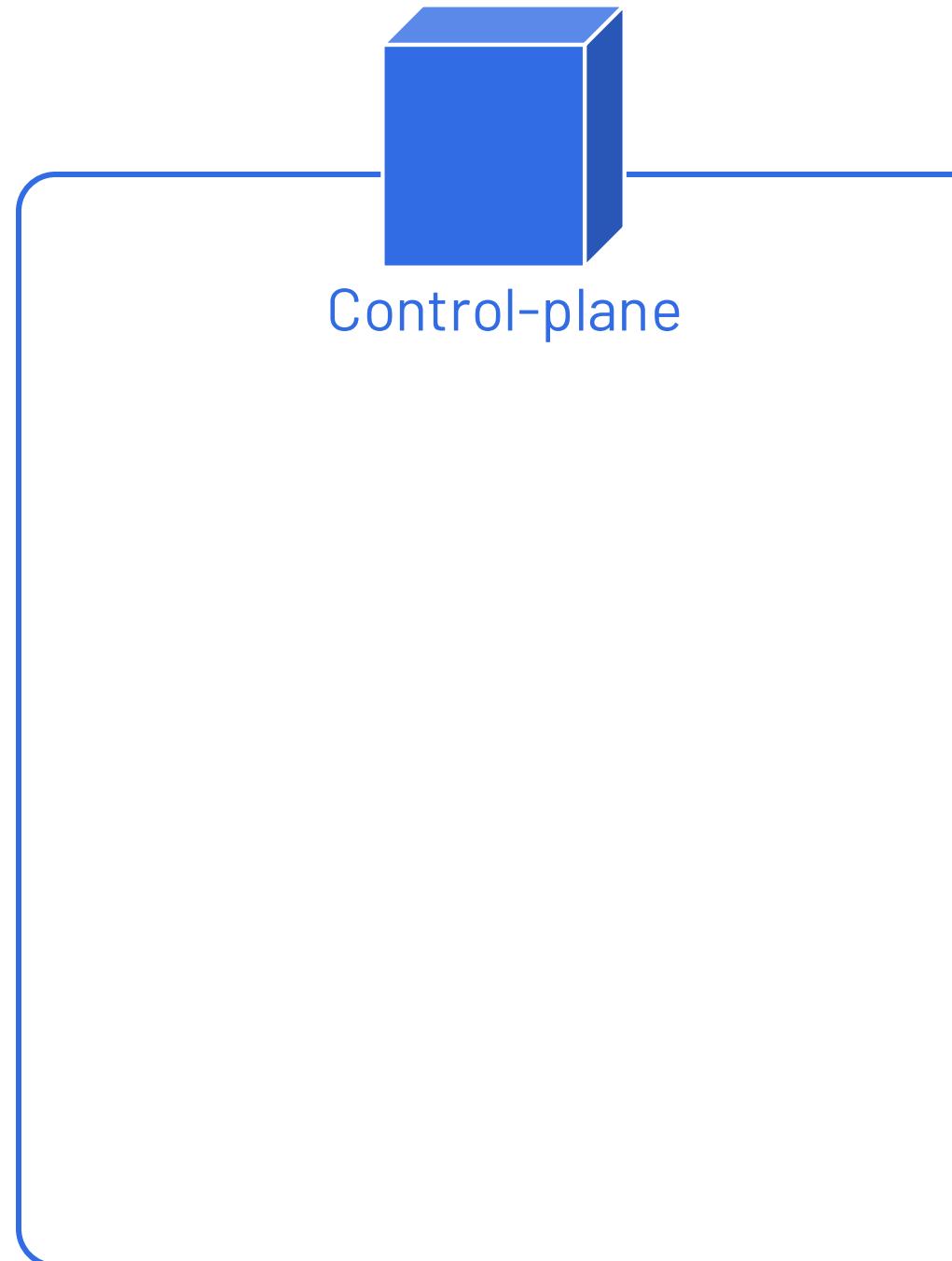
Control-plane

Central nervous system of the cluster

Manage, plan, schedule and monitor nodes and application containers

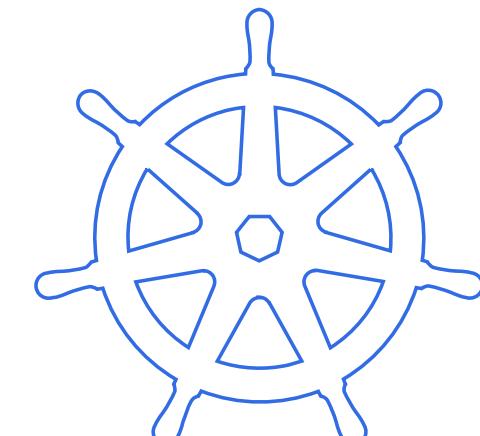


Control-plane Components

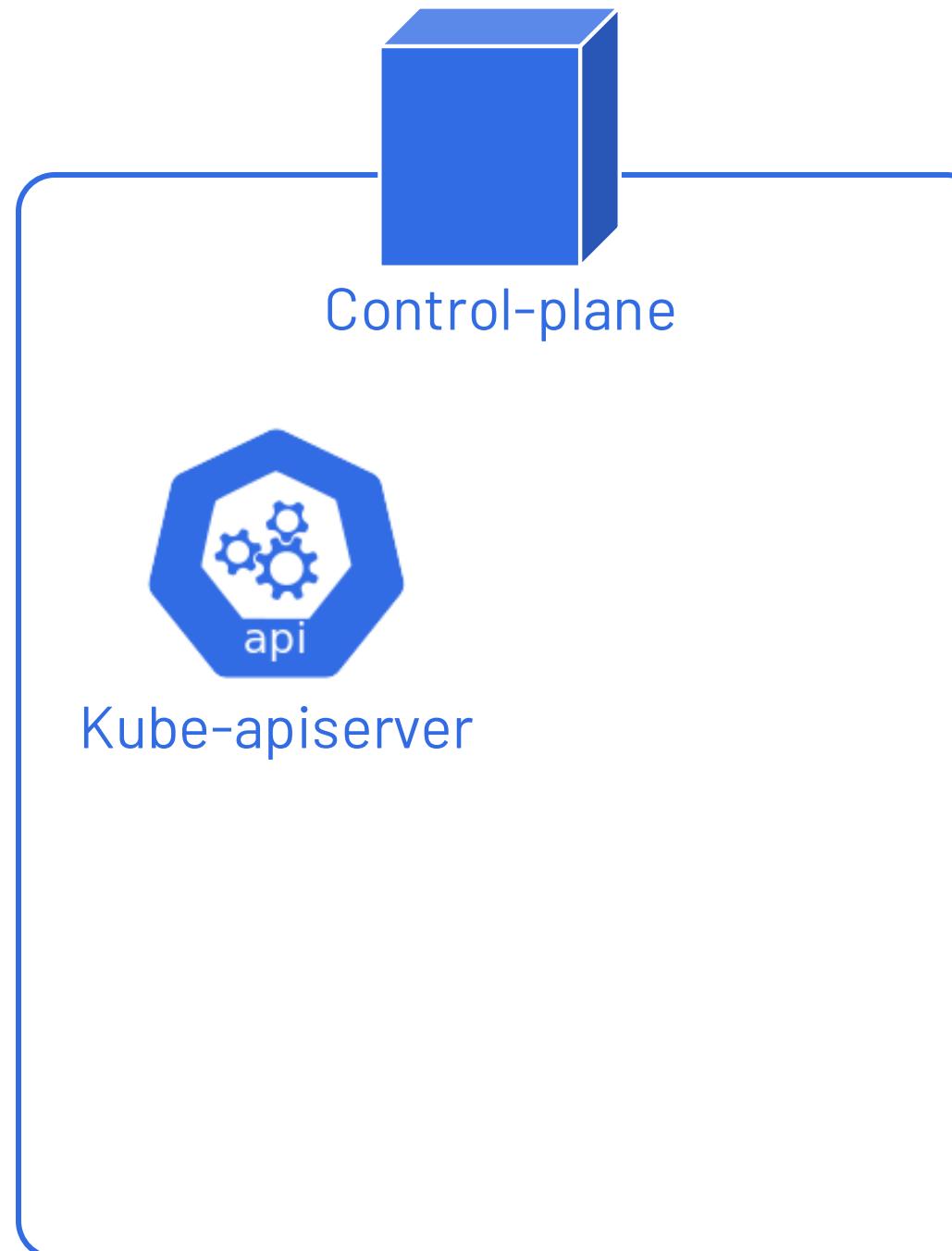


Kube-apiserver

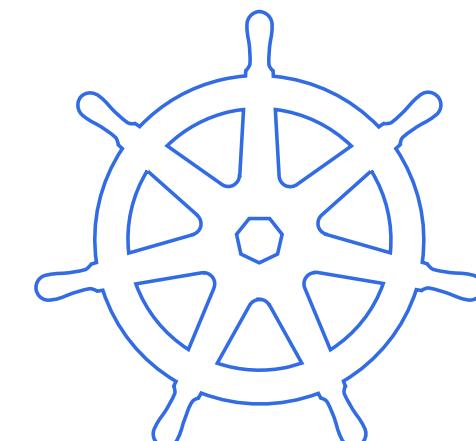
- ✓ Acts as a central point of communication & management
- ✓ Responsible for exposing K8s API's & orchestrating all operations within the cluster
- ✓ Serves as single point of entry for all user requests & interactions with the cluster



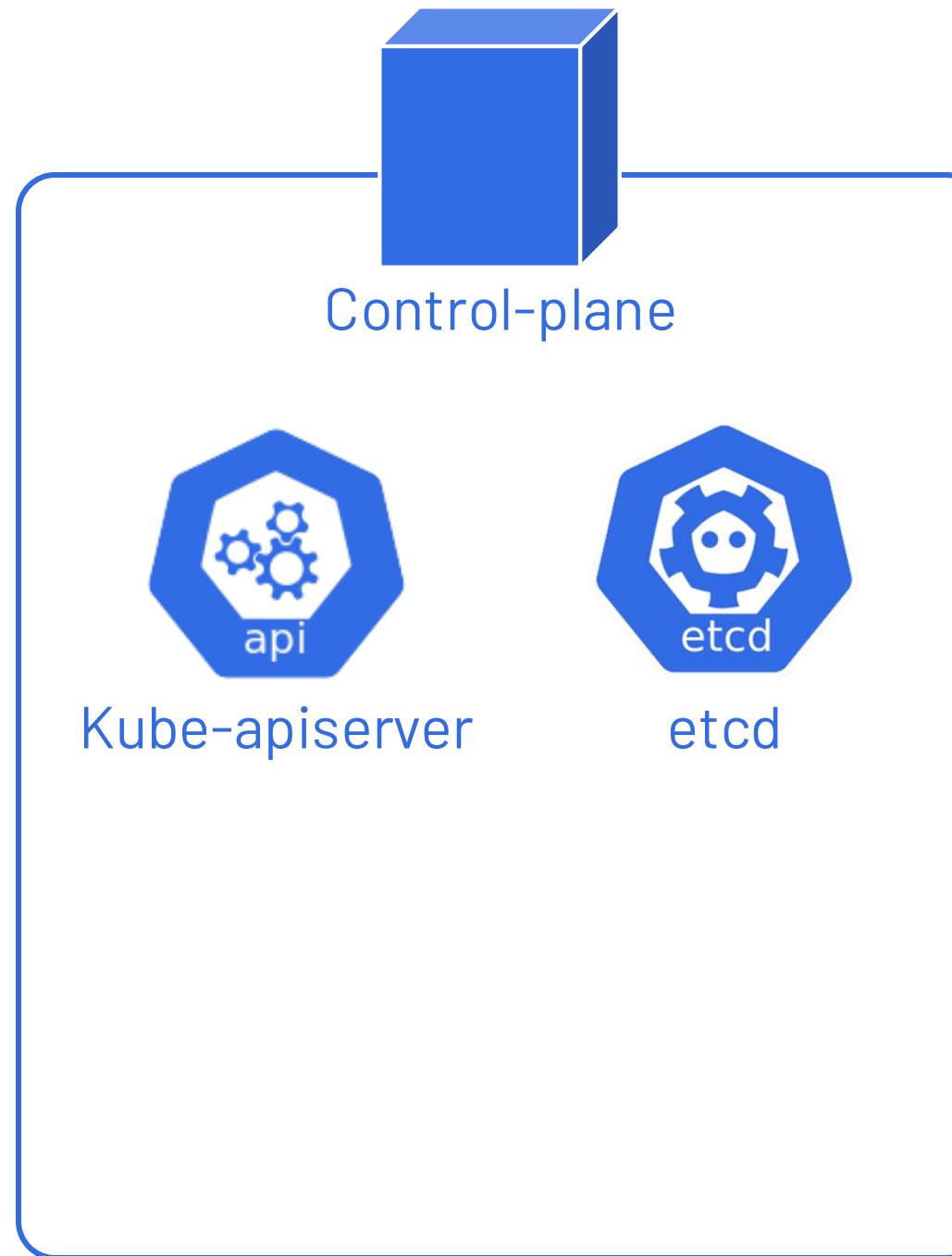
Control-plane Components



- ✓ Brain's memory for Kubernetes
- ✓ Securely stores all cluster configuration data & current state of all resources within the cluster
- ✓ Provides a single source of truth

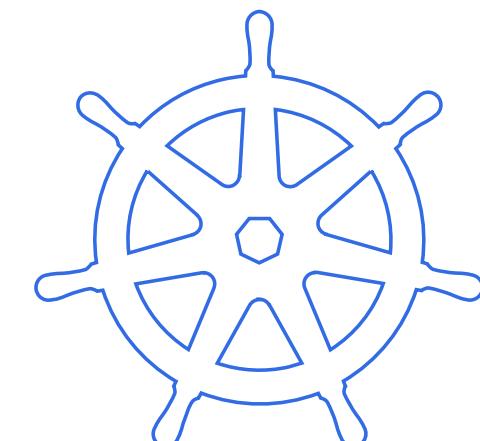


Control-plane Components

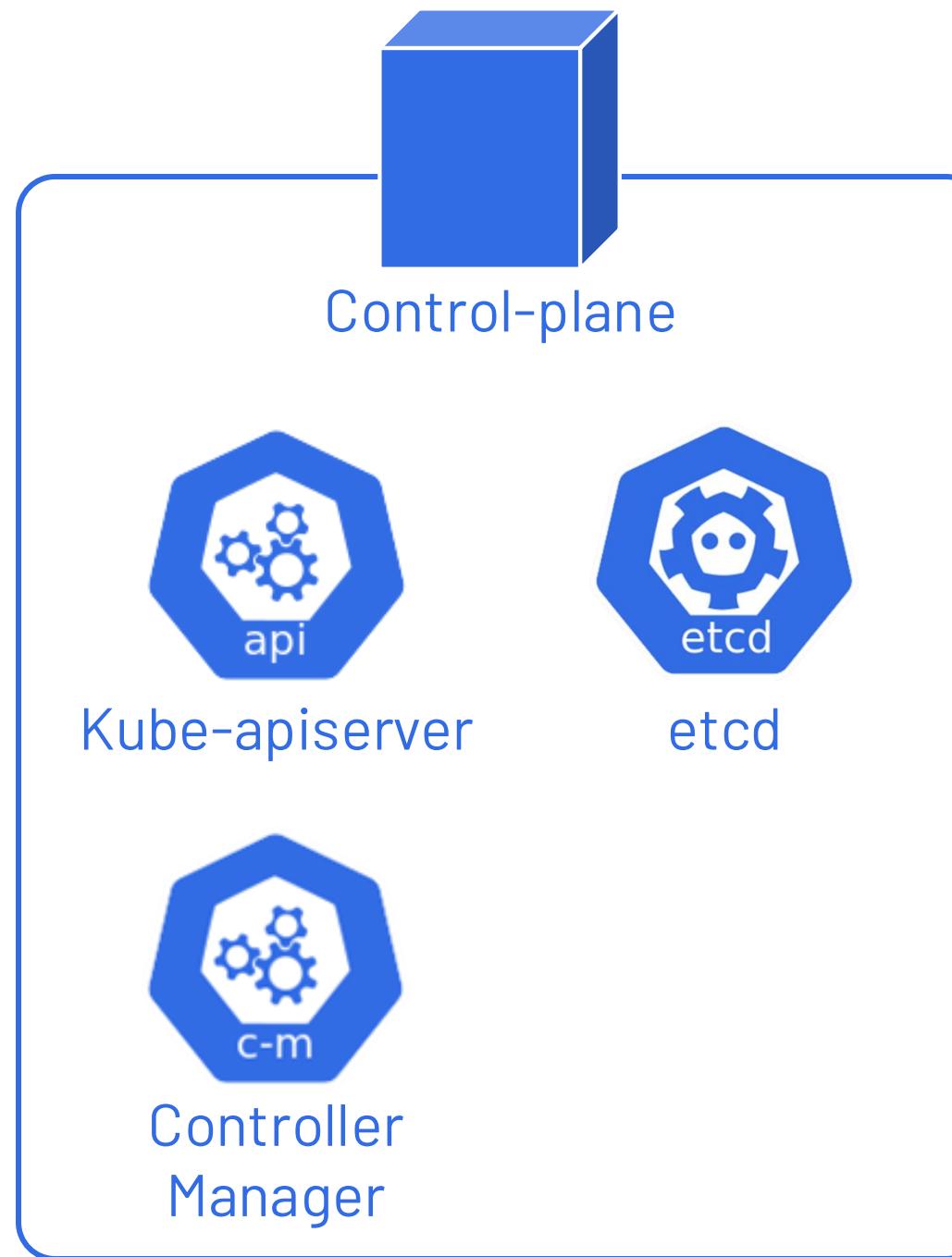


Controller Manager

- ✓ Diligent conductor of the Kubernetes
- ✓ Monitors current cluster state & compares it to desired state
- ✓ Orchestrates various specialized controllers, each responsible for a specific task

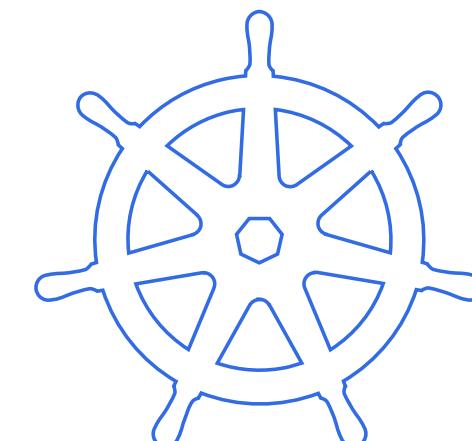


Control-plane Components

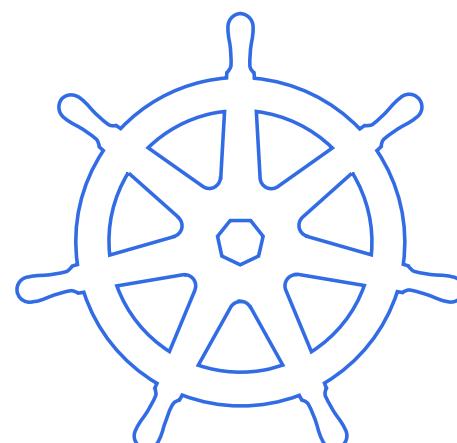
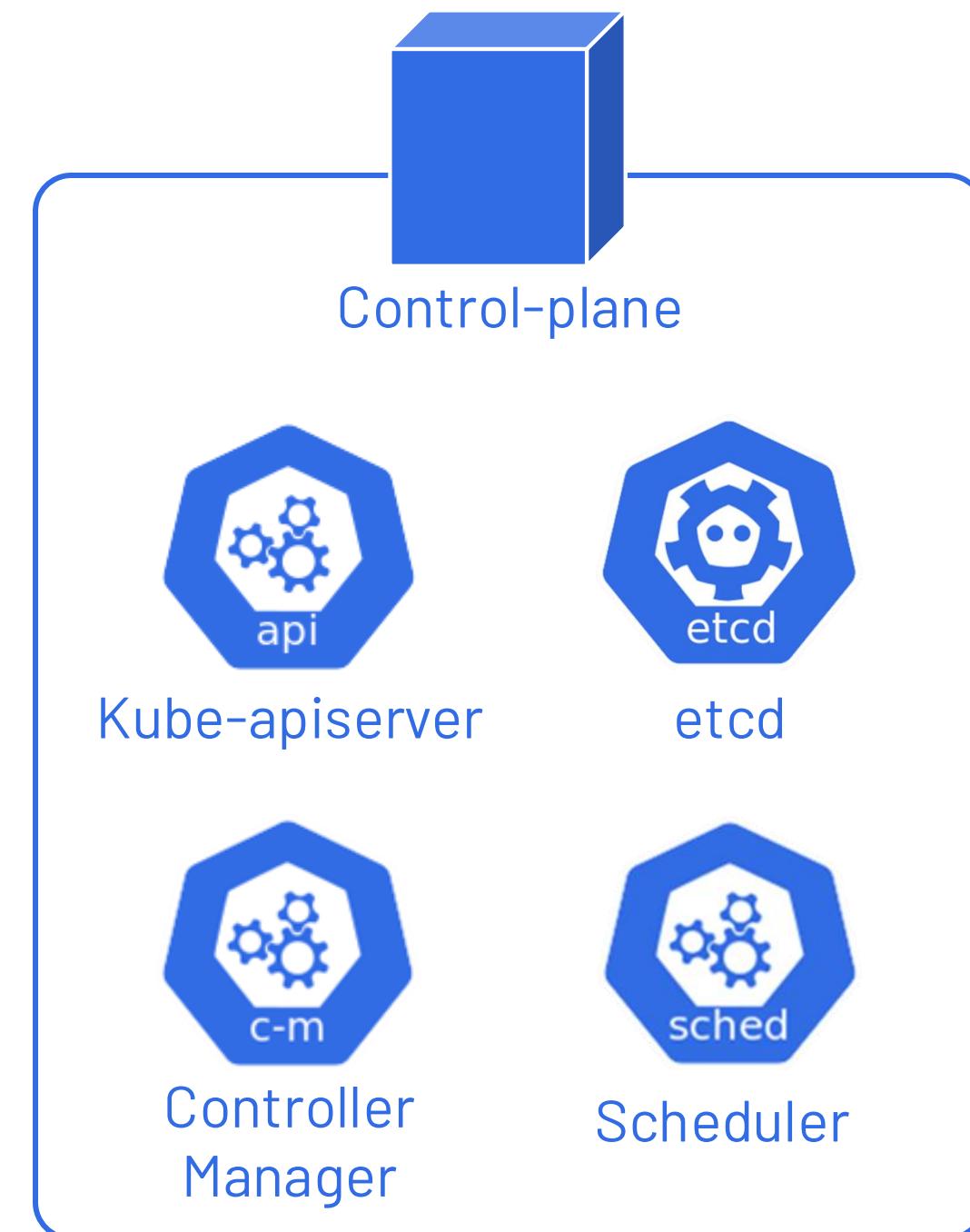


Scheduler

- ✓ Acts as the placement officer
- ✓ Analyzes factors of nodes
- ✓ On the basis of analysis, it makes informed decisions

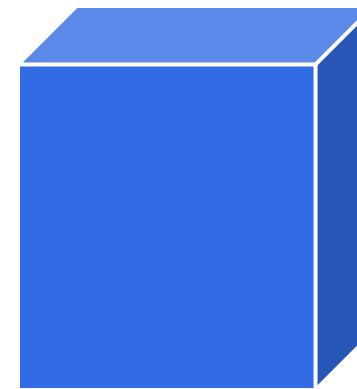


Control-plane Components



Node Components

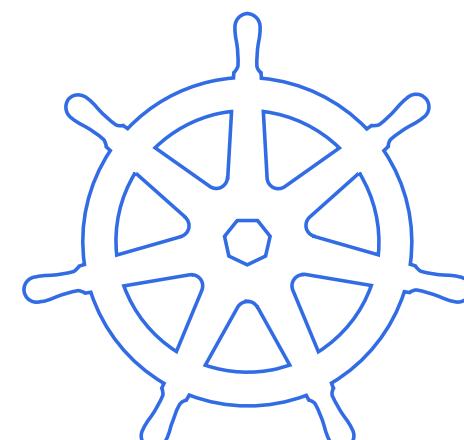
Node Components



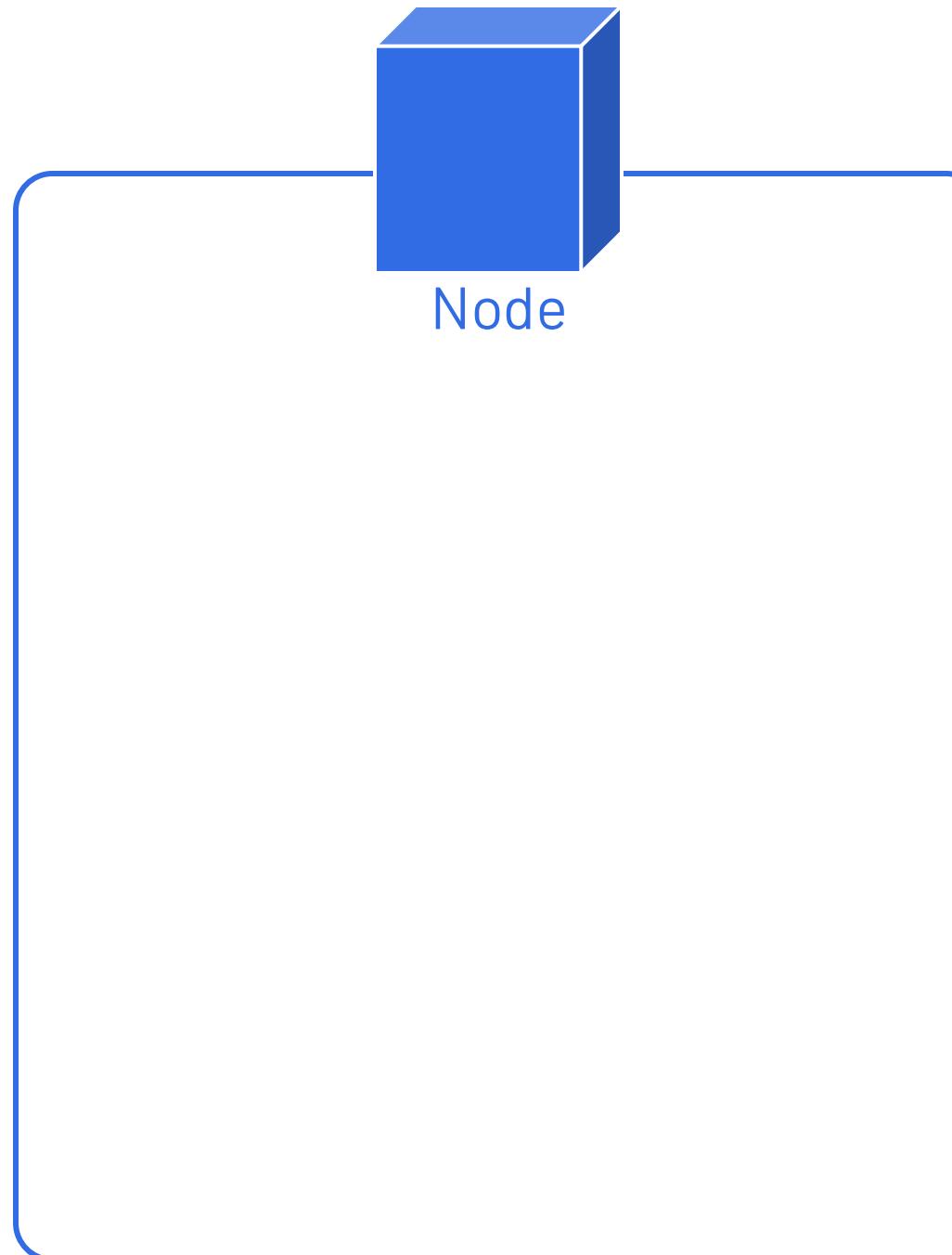
Node

Node components reside on every node in a cluster

Act as the workhorses that execute containerized applications

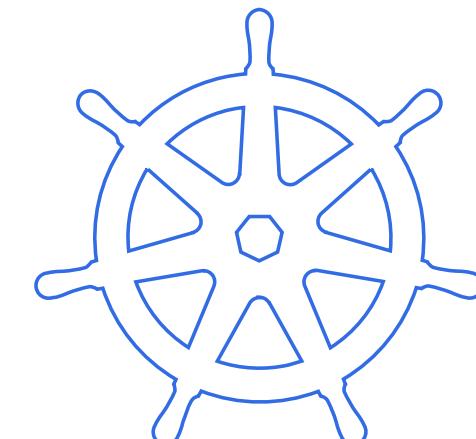


Node Components

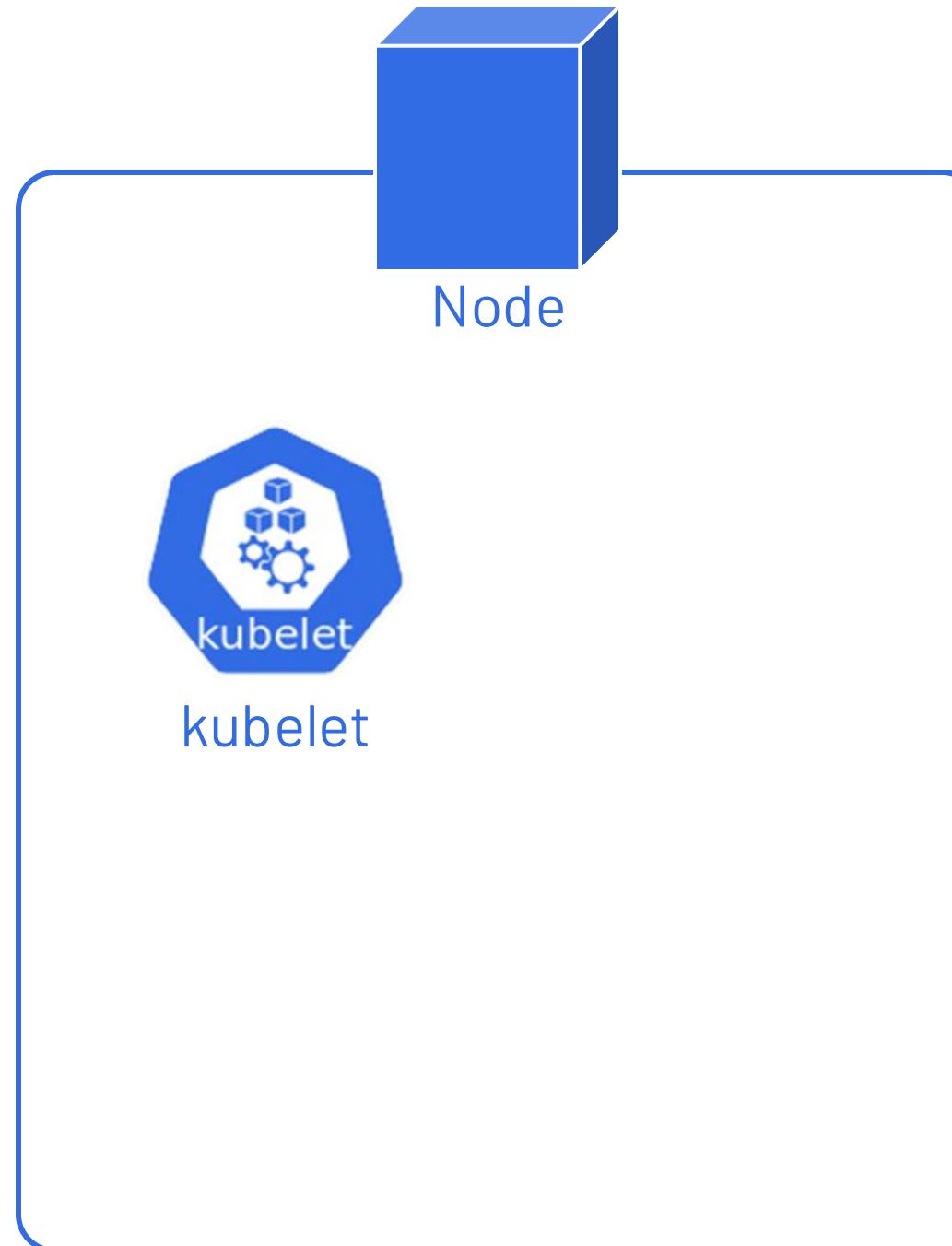


kubelet

- ✓ Functions as a captain on each node
- ✓ Receives instructions from kube-apiserver & manages lifecycle of pods
- ✓ Takes care of tasks like pulling container images from registries, allocating POD resources, reporting the health status

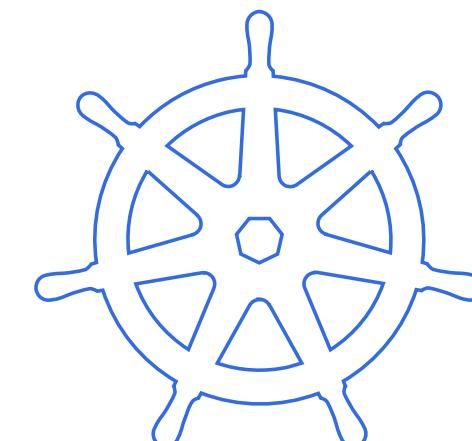


Node Components

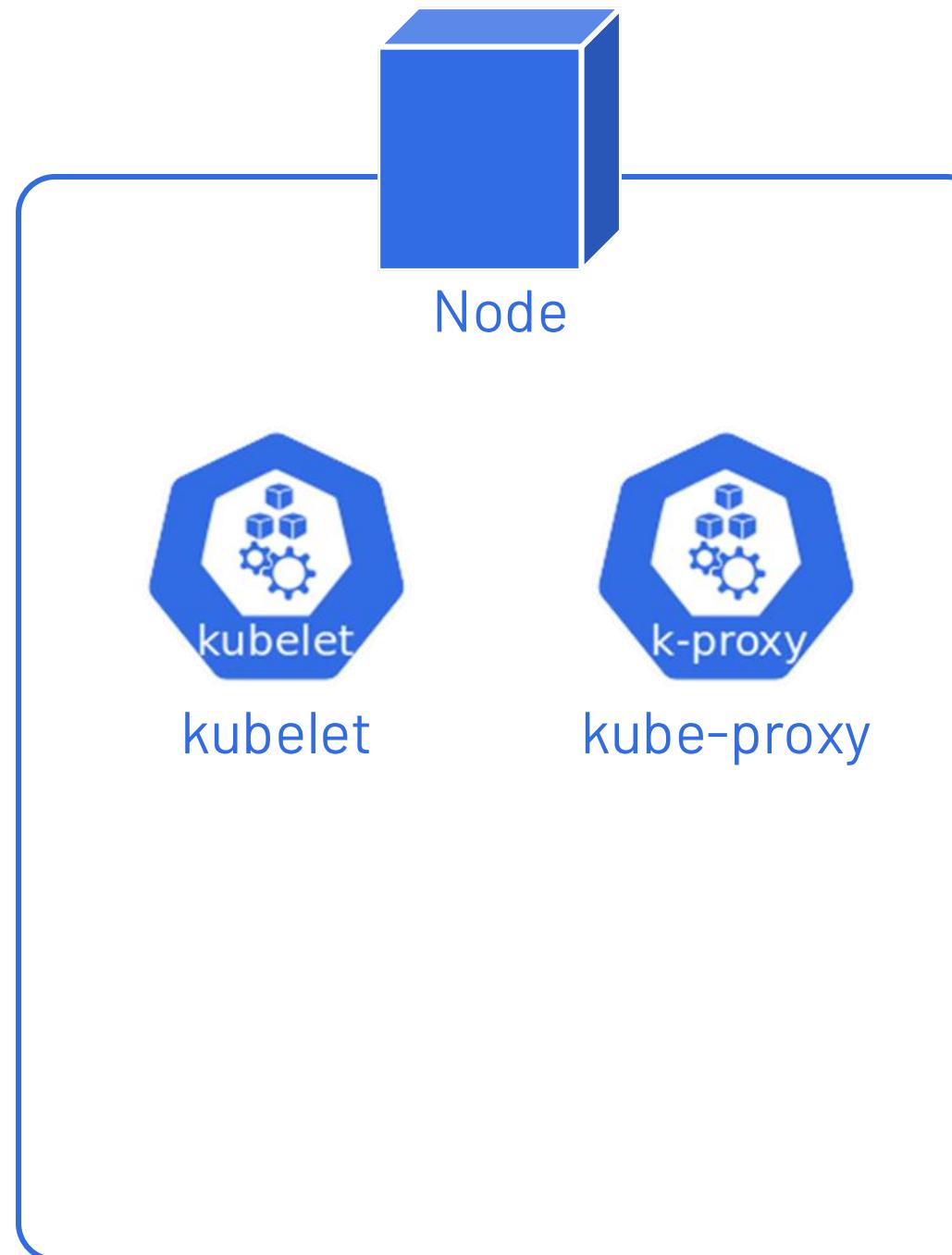


kube-proxy

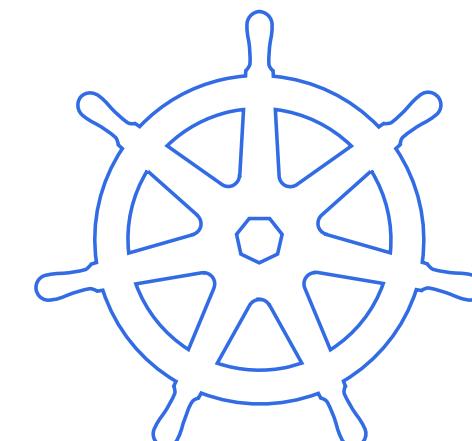
- ✓ Acts as the network traffic director
- ✓ Ensures pods can communicate with each other seamlessly
- ✓ Facilitates service discovery, allowing pods to locate & connect with services running within the cluster



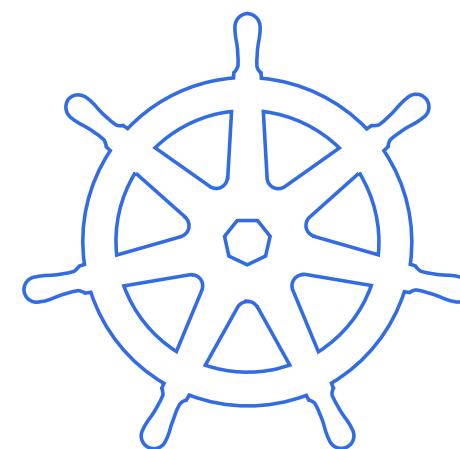
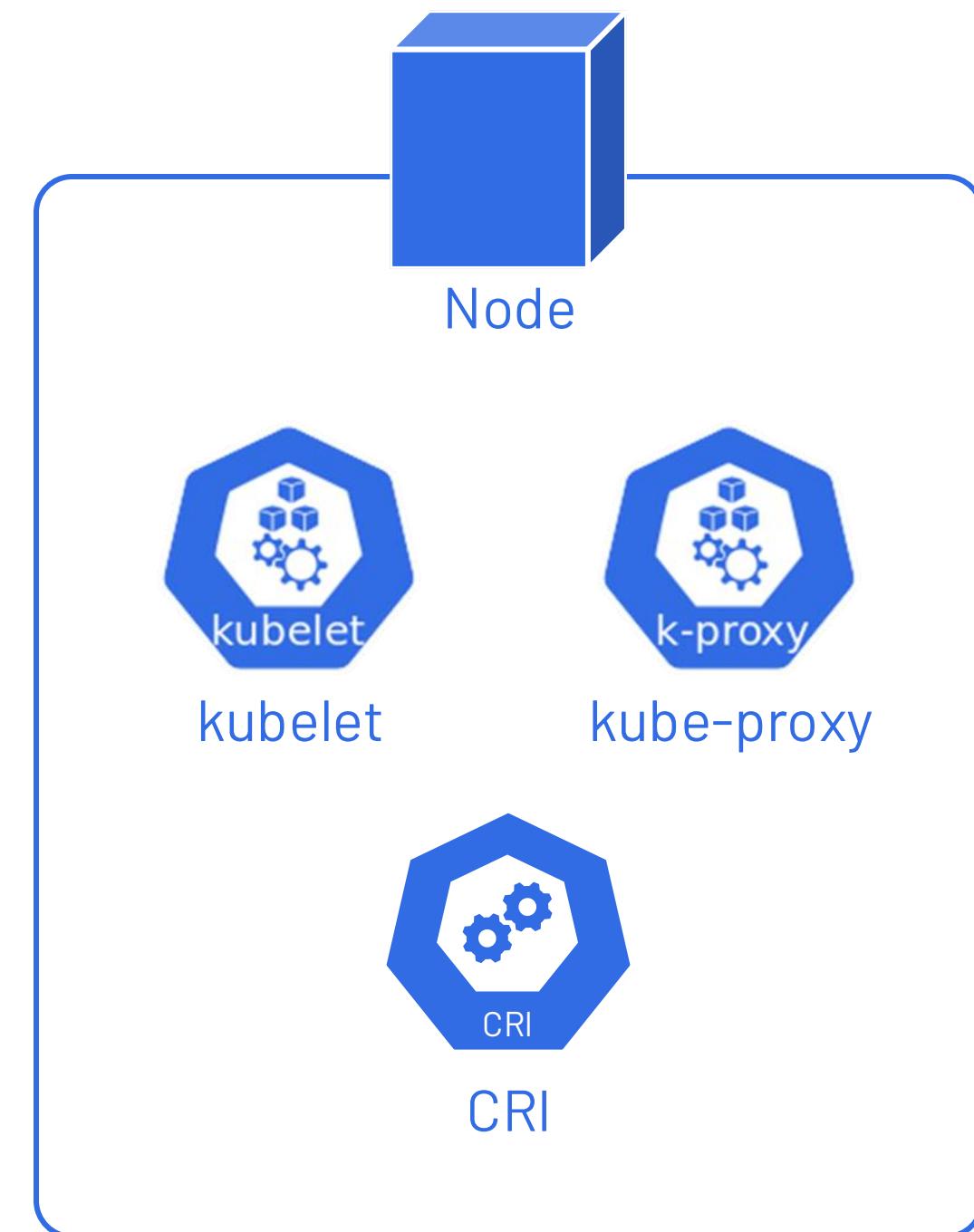
Node Components



- ✓ Act as the engine that powers the containers
- ✓ Popular options - Docker, containerd, and CRI-O
- ✓ Responsible for managing containers operations on the node



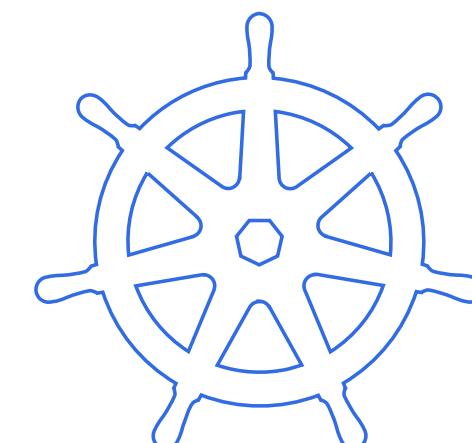
Node Components



Addons



- ✓ Network Plugins
- ✓ Service discovery addons
- ✓ State metrics
- ✓ Kubernetes dashboard



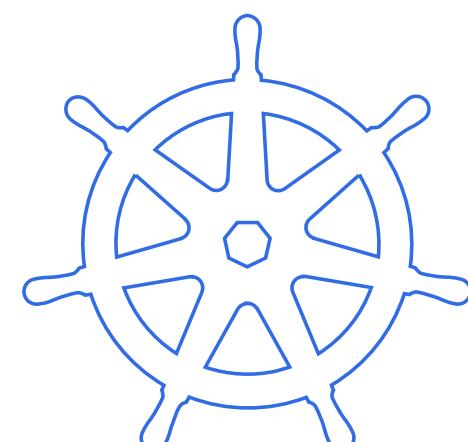
Two Node Cluster



Control Plane/ Master Node



Worker Node



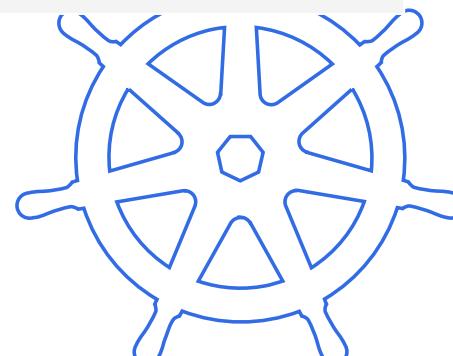
Pre-Requisites (Master Node)

Pre-Requisites (Master Node)



- ✓ Linux Host
- ✓ Memory - 4 GB RAM or more per node
- ✓ CPU - 2 CPUs or more
- ✓ Disable Swap
- ✓ Disable SELinux
- ✓ Network Connectivity
- ✓ Unique Identifiers (hostname or MAC address)
- ✓ Port Requirements

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443	Kubernetes API server	All
TCP	Inbound	2379-2380	etcd server client API	kube-apiserver, etcd
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	10259	kube-scheduler	Self
TCP	Inbound	10257	kube-controller-manager	Self
TCP	Inbound	10256	kube-proxy	Self, Load balancers



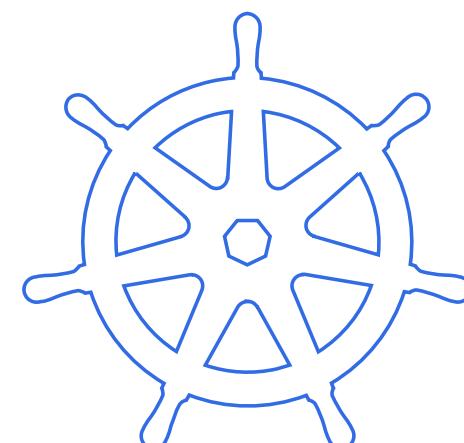
Pre-Requisites (Worker Node)

Pre-Requisites (Worker Node)



- ✓ Linux or Windows Host
- ✓ Memory - 2 GB RAM more per node
- ✓ CPU - 2 CPUs or more
- ✓ Disable Swap
- ✓ Disable SELinux
- ✓ Network Connectivity
- ✓ Unique Identifiers (hostname or MAC address)
- ✓ Port Requirements

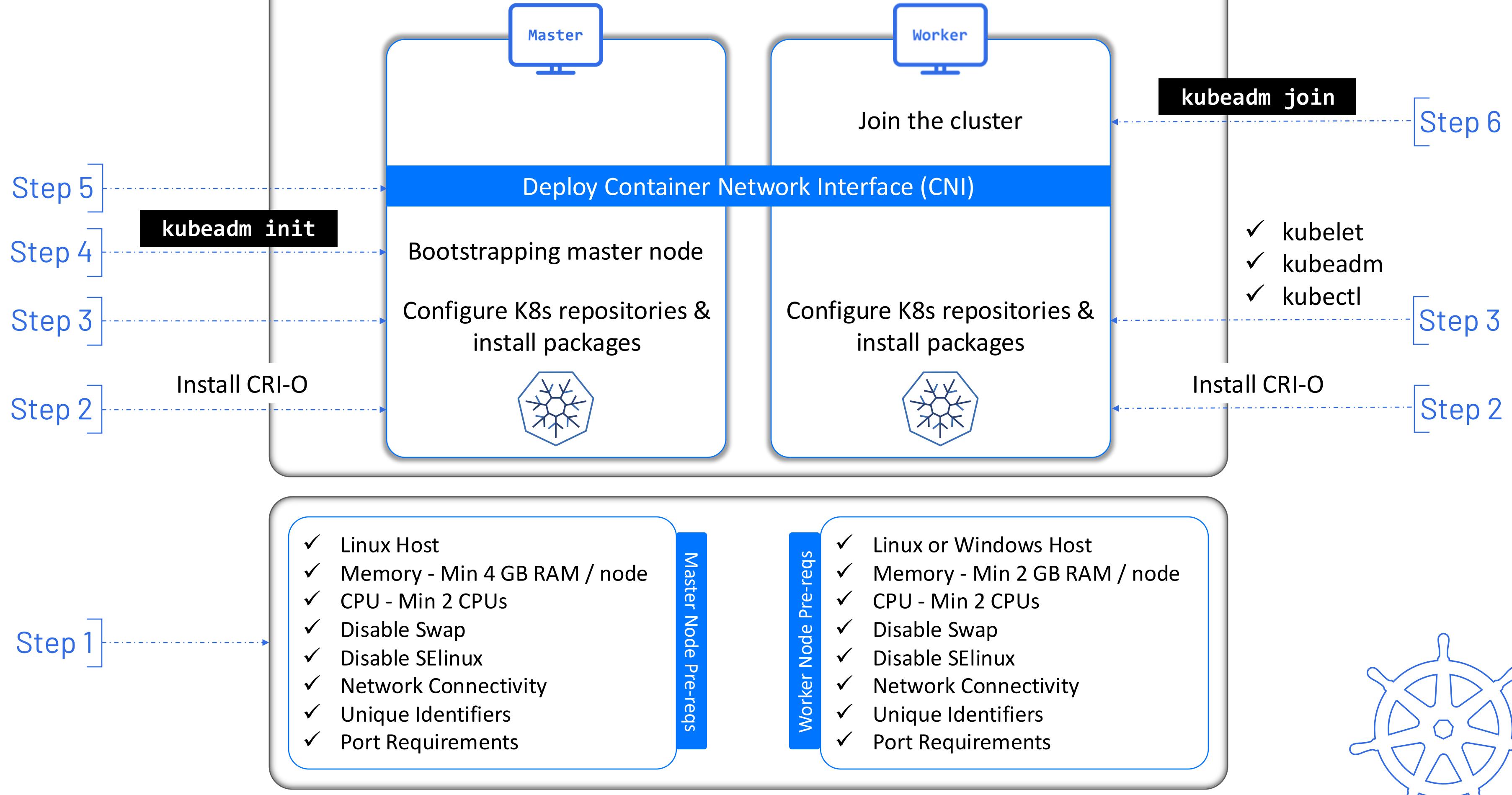
Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	10256	kube-proxy	Self, Load balancers
TCP	Inbound	30000-32767	NodePort Services	All



Setting up Two-node Cluster

Two Node K8s Cluster

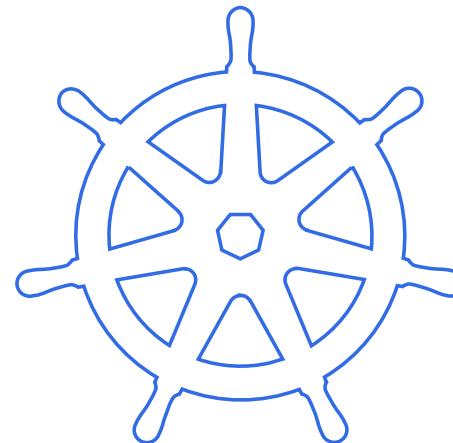
kubectl get nodes





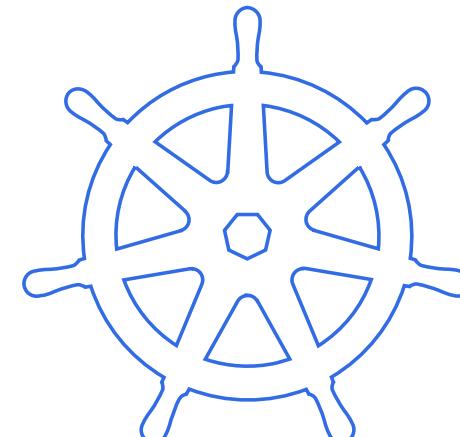
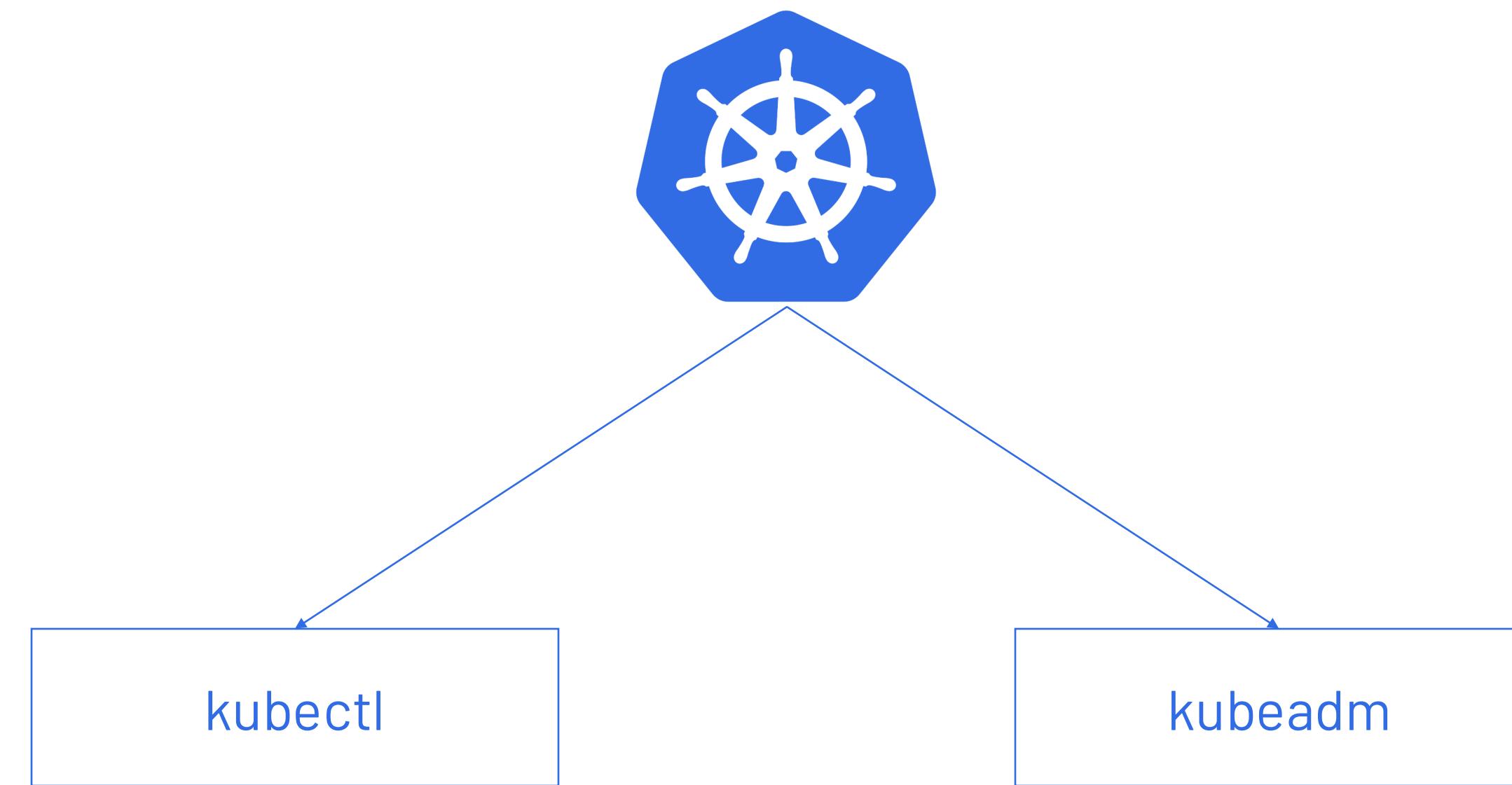
Demo

Two Node
Cluster Setup



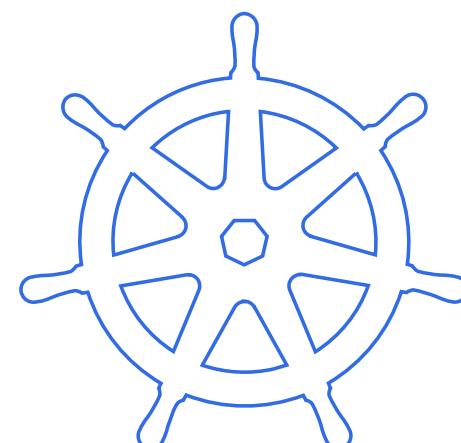
Kubernetes CLI Overview

Kubernetes CLI Overview



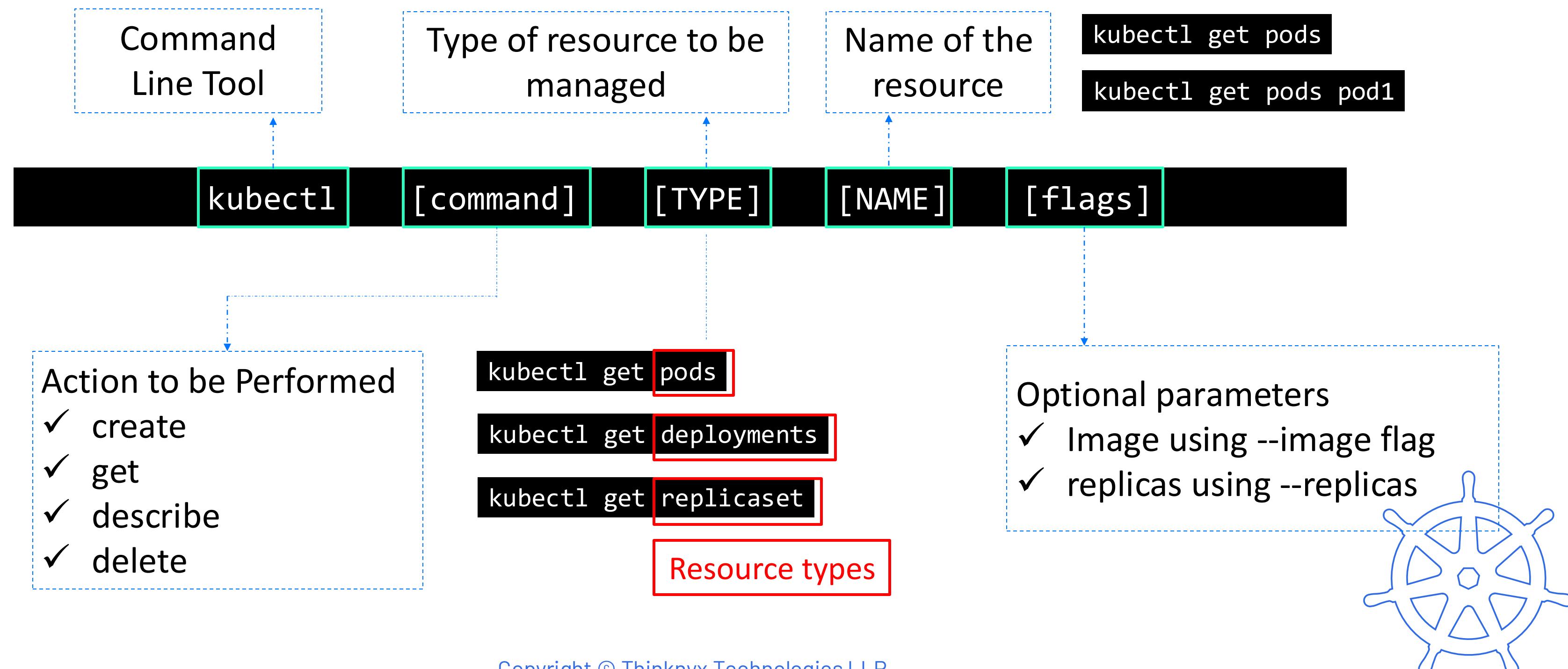
kubectl

- Command-line tool - Communicates with control plane of K8s cluster via K8s API
 - ✓ Create
 - ✓ Update
 - ✓ Delete
 - ✓ Obtain Information

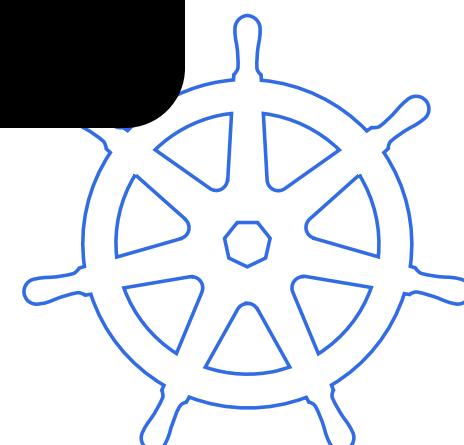
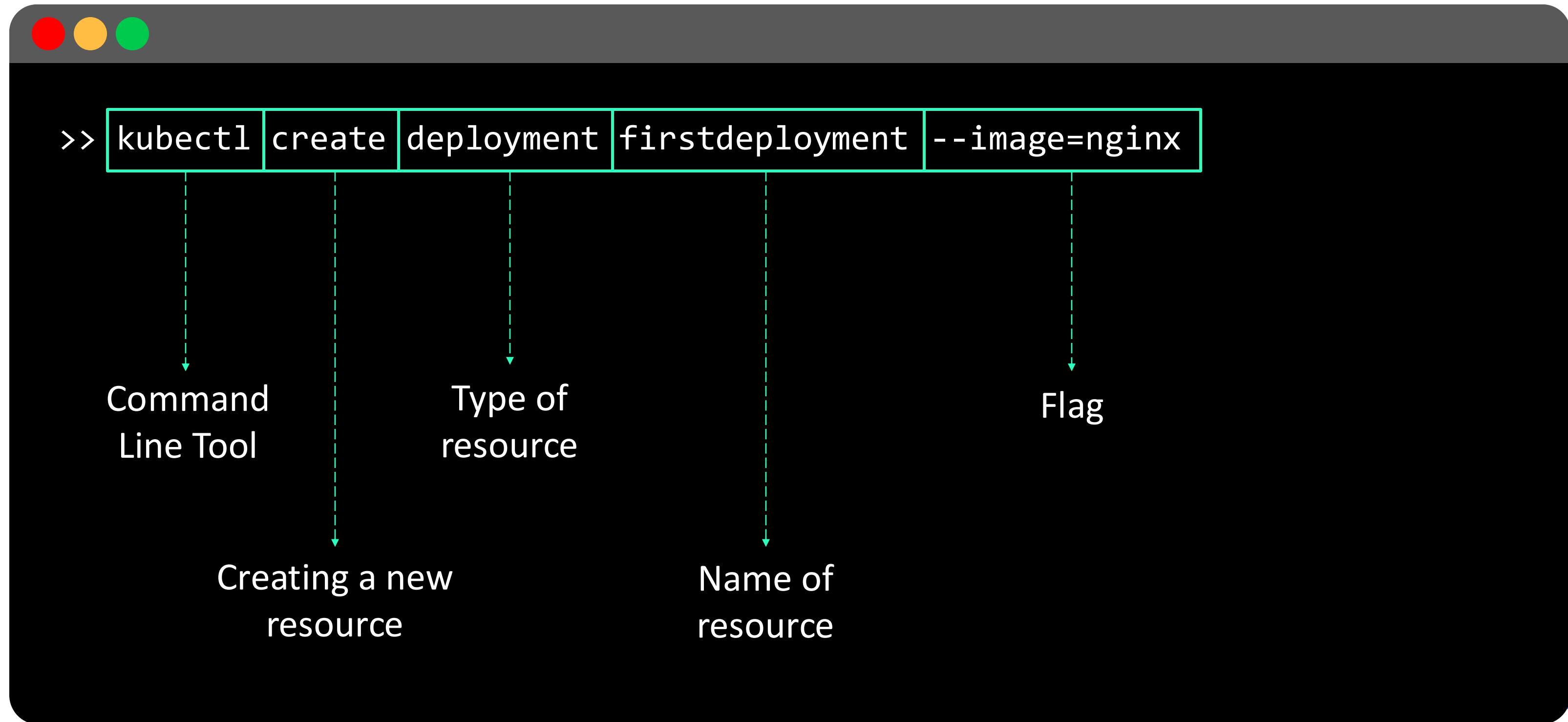


kubectl

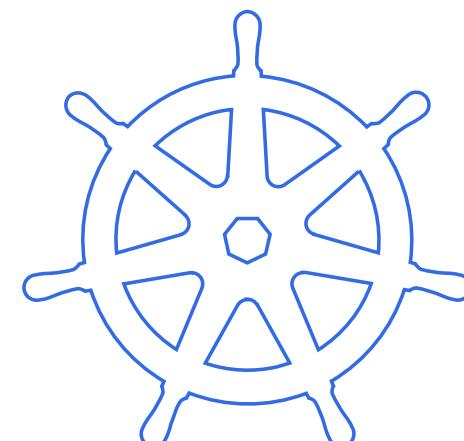
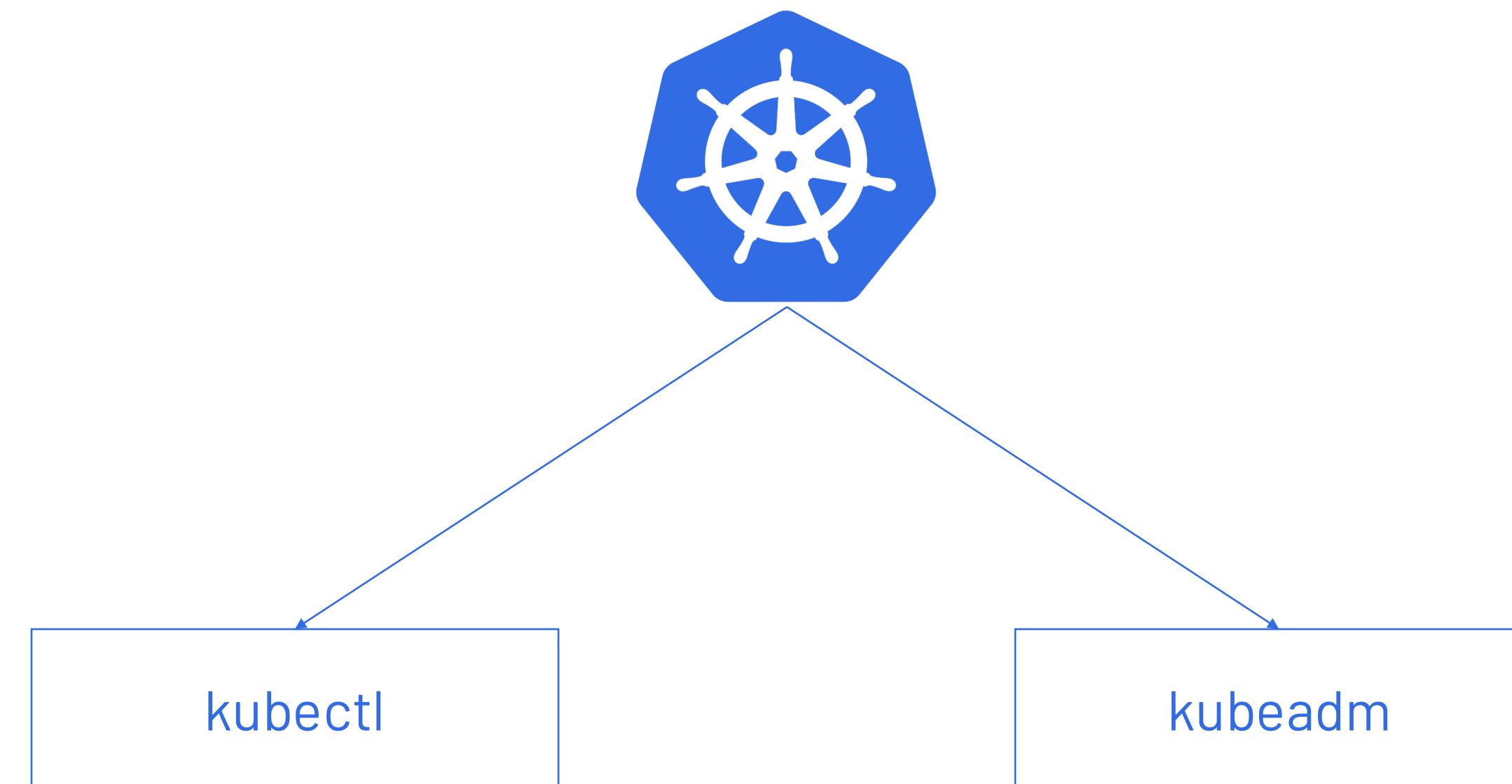
Basic syntax of kubectl command



kubectl

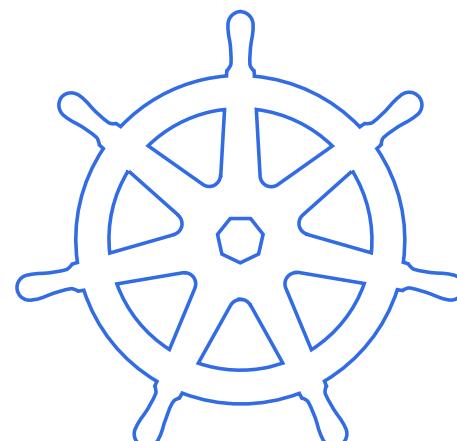


Kubernetes CLI Overview



kubeadm

- Used to build K8s clusters
- Focuses on bootstrapping rather than provisioning machines or installing additional add-ons
 - ✓ Upgrading clusters
 - ✓ Managing tokens
 - ✓ Resetting configurations
 - ✓ Handling certificates
 - ✓ Managing kubeconfig files



Kubernetes Objects

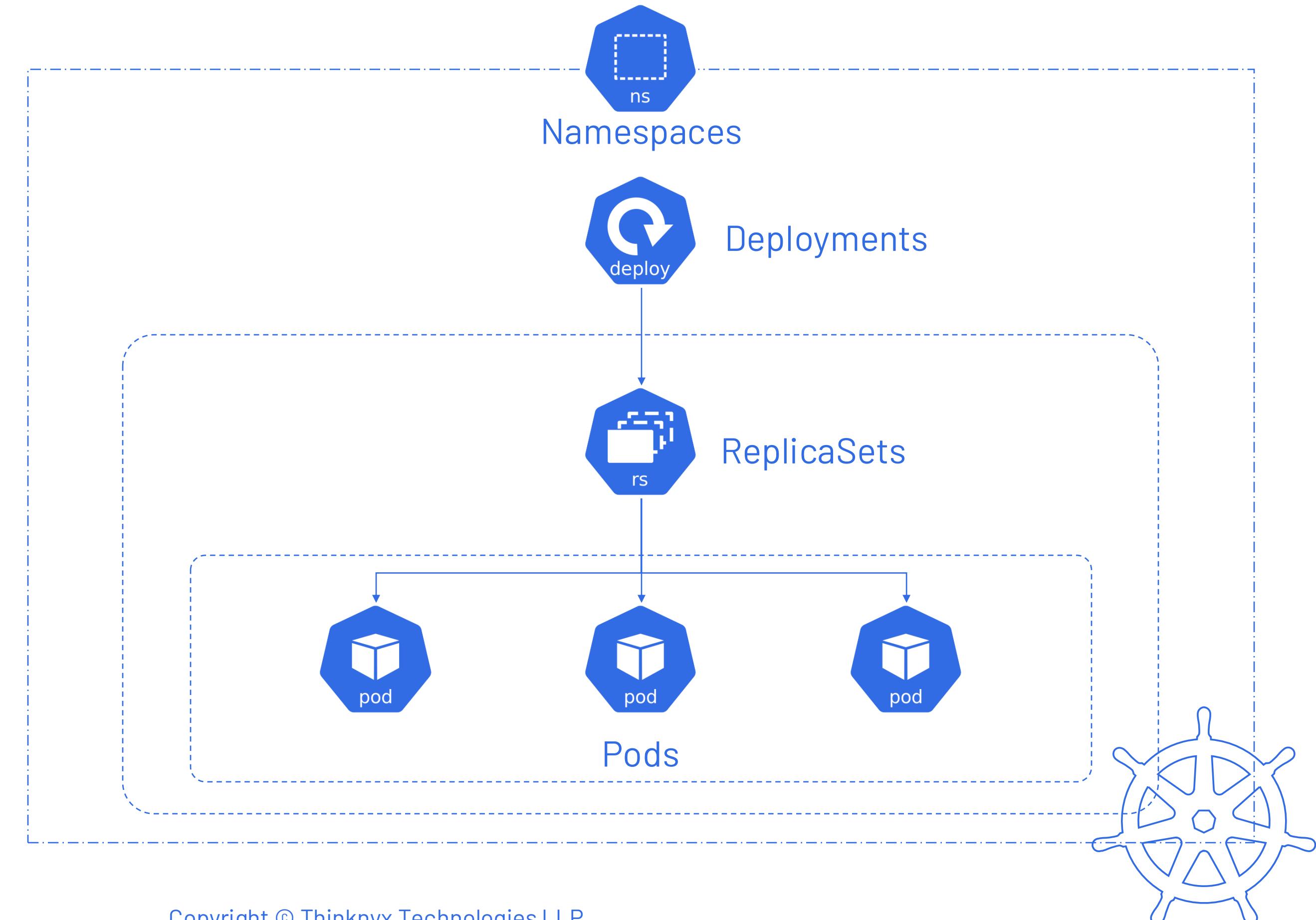
Kubernetes Objects

Provides a way to create logical partitions within a cluster

Manage Pod's lifecycle & provide advanced capabilities for application management

Ensure continuous pod availability by automatically replacing failed pods with new ones

Smallest installable units of computing that can be created and managed in K8s



Namespaces

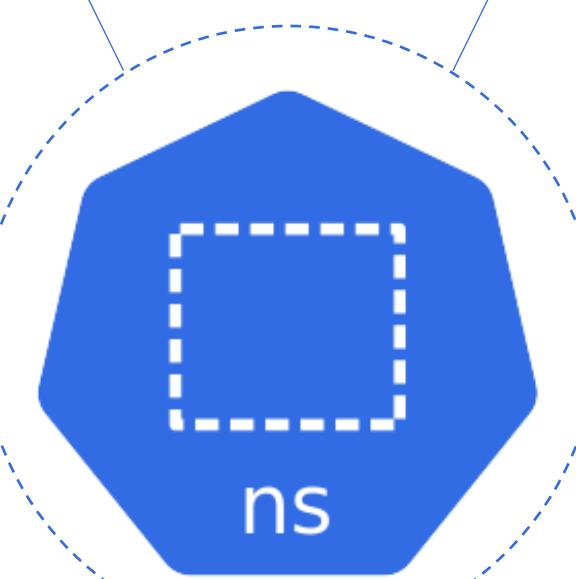
Namespaces

Default Namespaces

Creates four default namespaces

- default
- kube-node-lease
- kube-public
- kube-system

to manage cluster resources



Resource Isolation

Isolate resources in a cluster, preventing interference between users or teams

Resource Organization

Helps in organizing and managing resources specific to application or environment

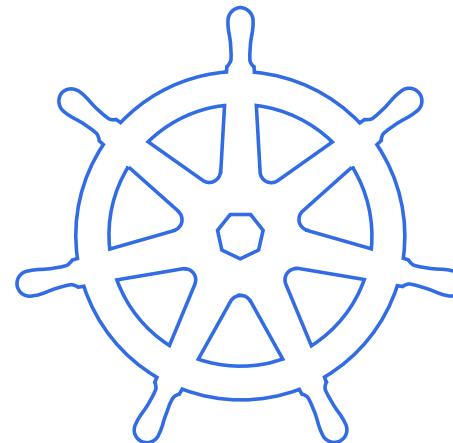
Resource Quotas

Enforce limits on resource usage to avoid excessive consumption by any single team





Demo | Namespaces



Pods

Pods

Ephemeral Nature

Dynamically created, destroyed, or replaced in response to scaling needs, resource limitations, or node failures

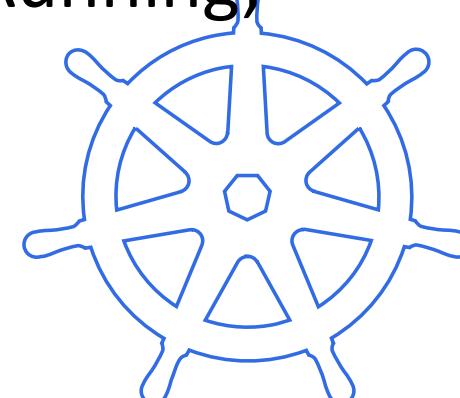


Group of Containers

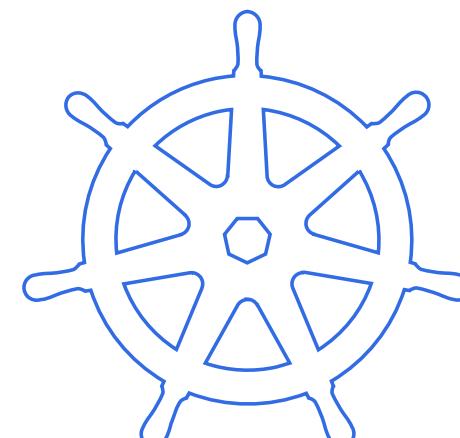
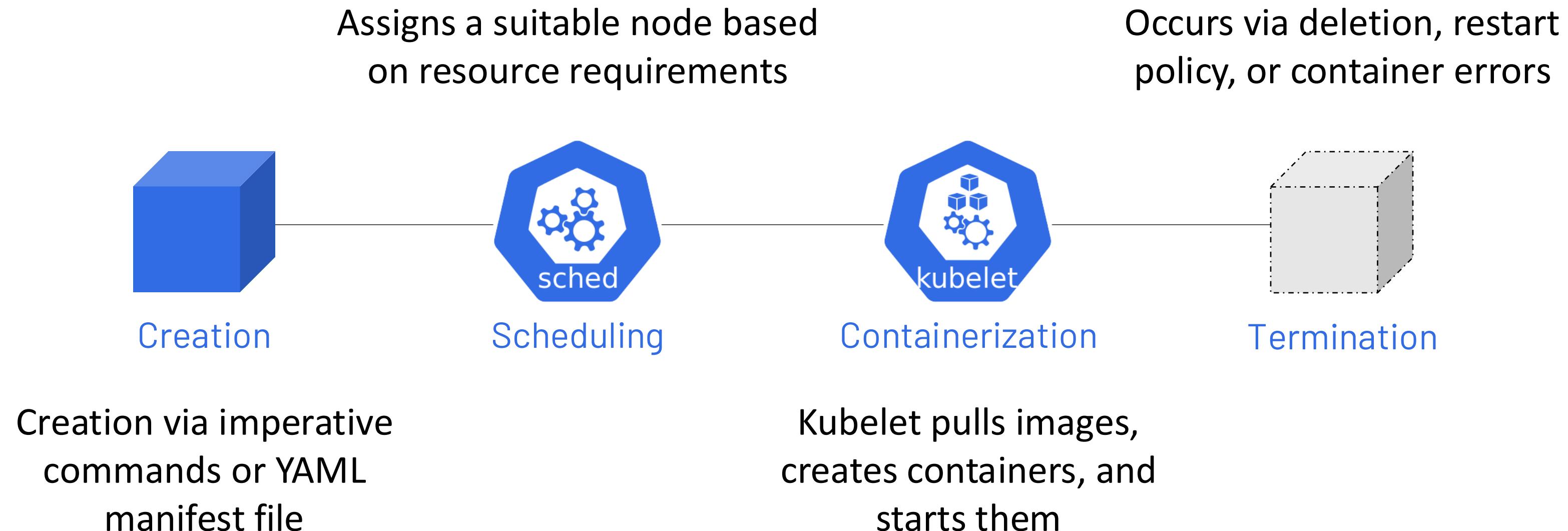
Groups of one or more tightly coupled containers that share the same network namespace and storage

Lifecycle

Go through phases like Pending, Running, Succeeded, or Failed

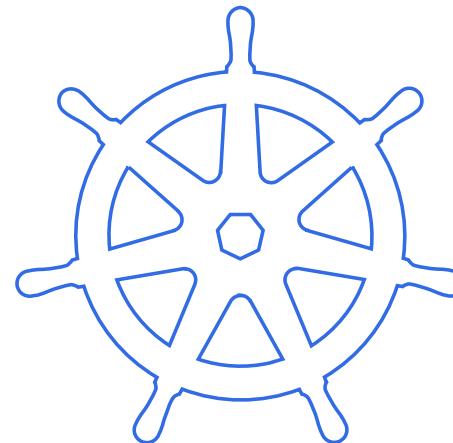


Pods



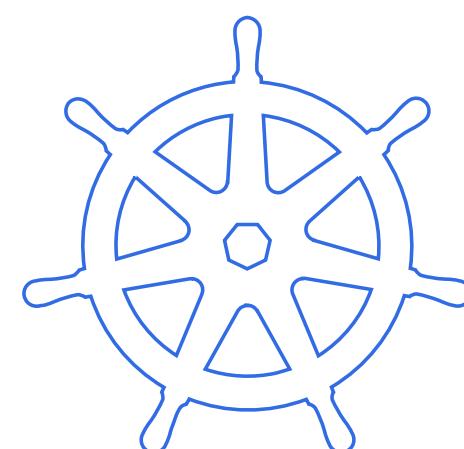
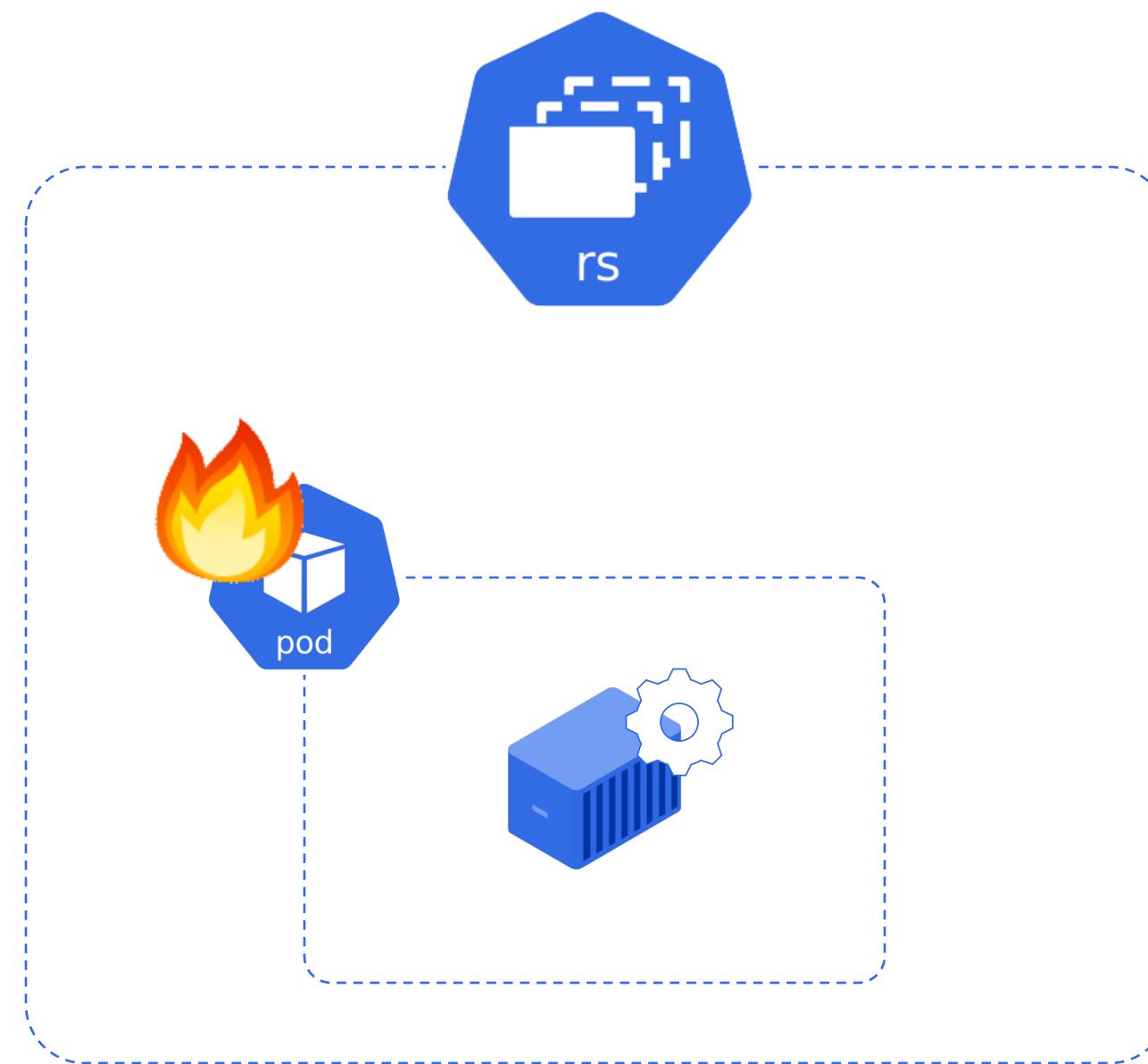


Demo | Pods



ReplicaSets

ReplicaSets

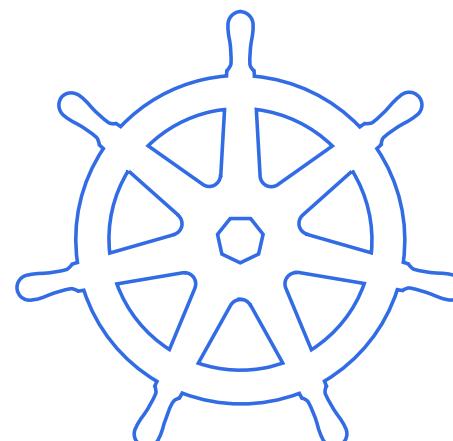


How ReplicaSets Works?

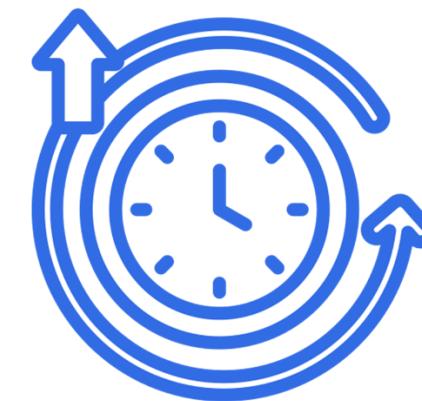
Define
ReplicaSet
Object

Monitor
running Pods

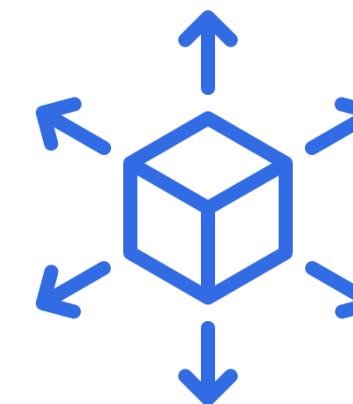
Effortless
Scaling



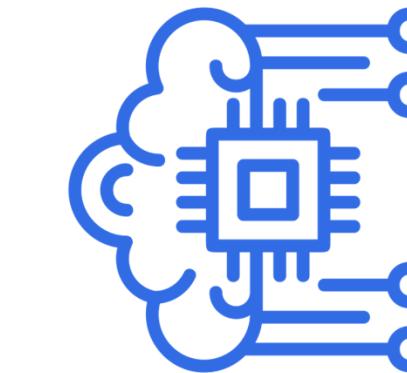
Benefits of ReplicaSets



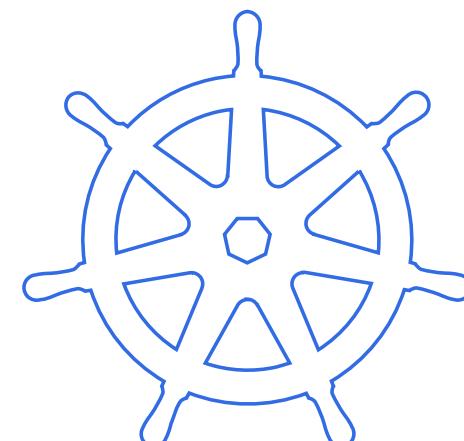
High Availability



Scalability

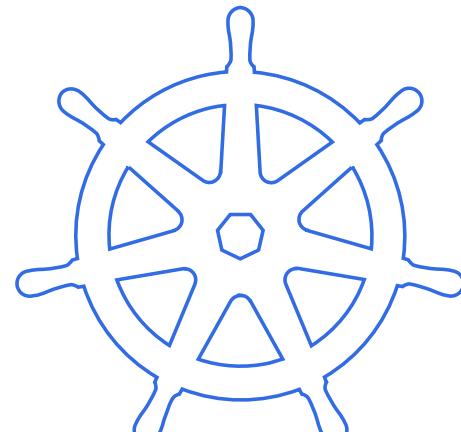


Self-healing





Demo | ReplicaSets



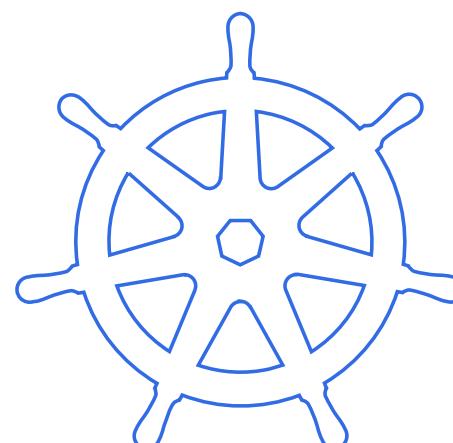
Deployments

Deployments



Offers advanced features such as rolling updates & rollbacks, makes it easier to manage the application lifecycle

Provides declarative updates to applications and ensures pods are running at a given time



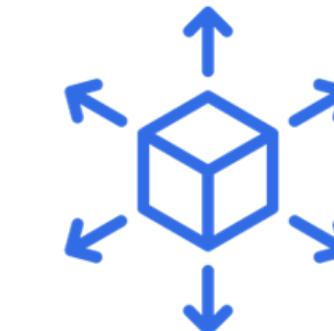
Benefits of Deployments



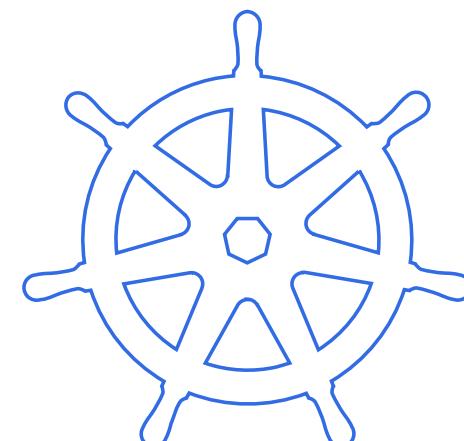
Update Strategies



Rollbacks

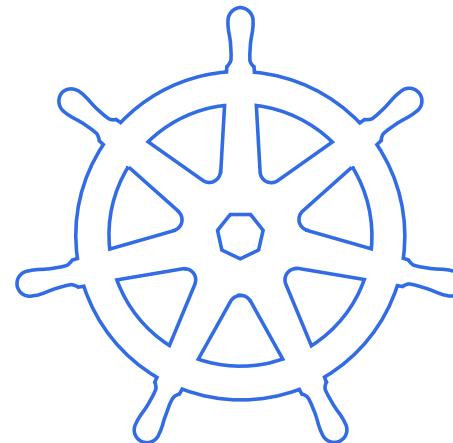


Scaling





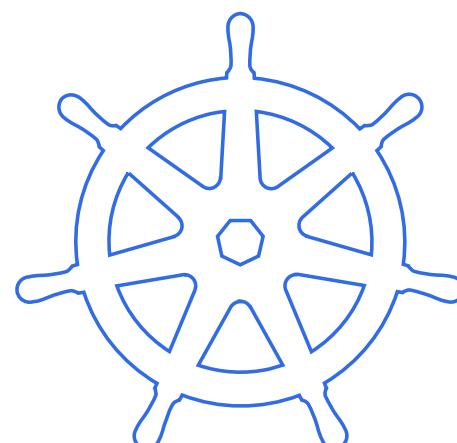
Demo | Deployments



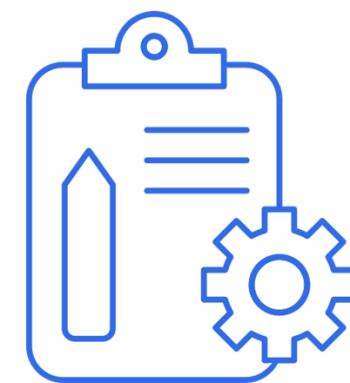
Labels & Selectors

Labels & Selectors

- Provides a way to categorize & target resources within a cluster
 - ✓ Labels categorize resources in Kubernetes with identifying tags
 - ✓ Selectors filter resources based on their assigned labels



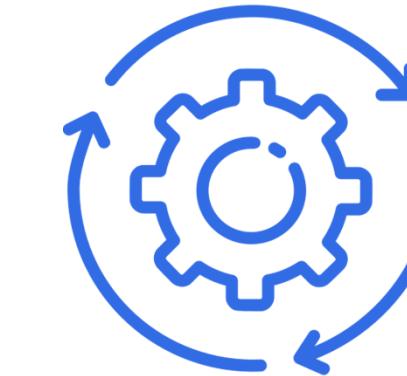
Benefits of Labels & Selectors



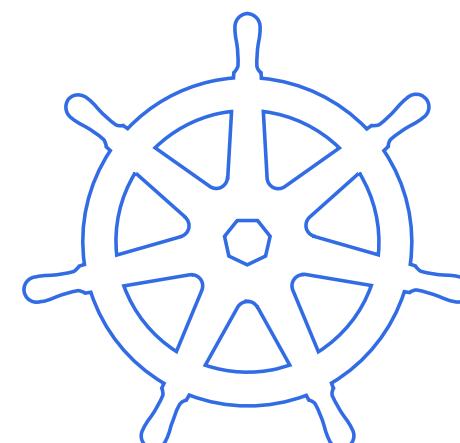
Categorization
/Organization



Selection



Automation



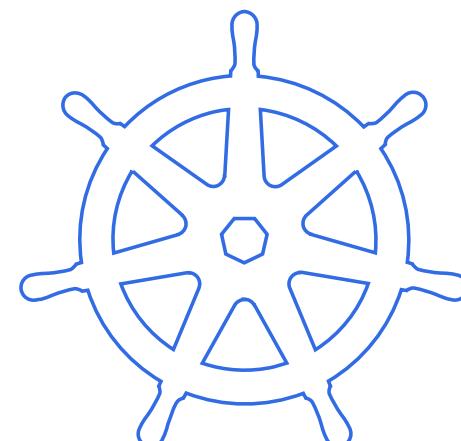
Key Concepts

Key-value pairs for classifying
Kubernetes resources

Labels

Selectors

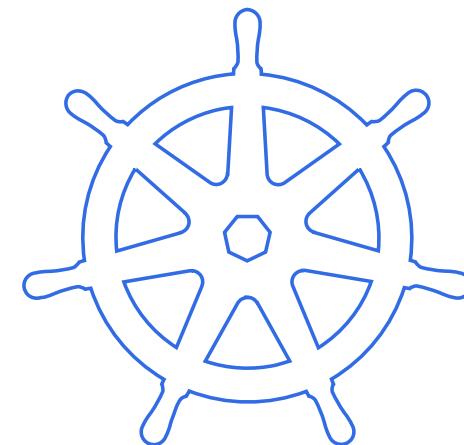
Filter resources based on labels
key-value comparisons





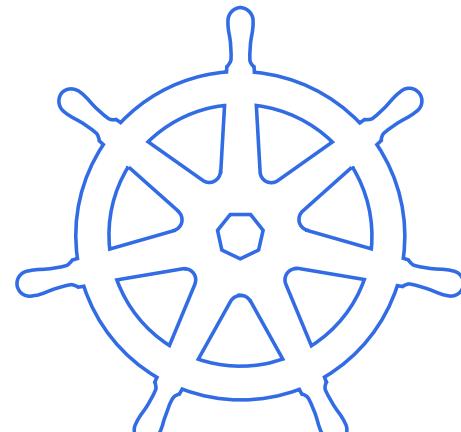
Demo

Labels &
Selectors





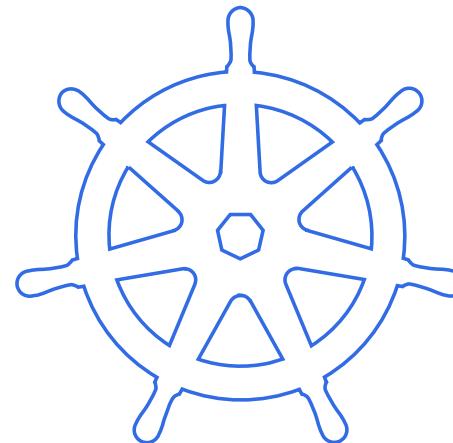
Demo | Inbuilt Labels





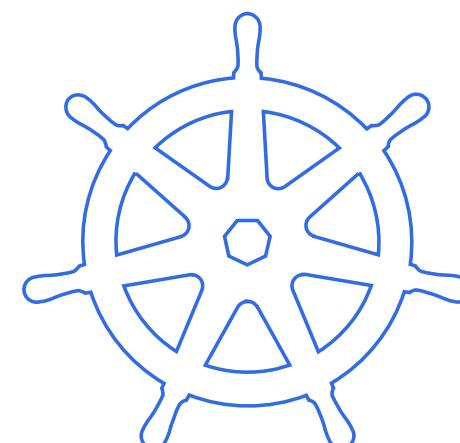
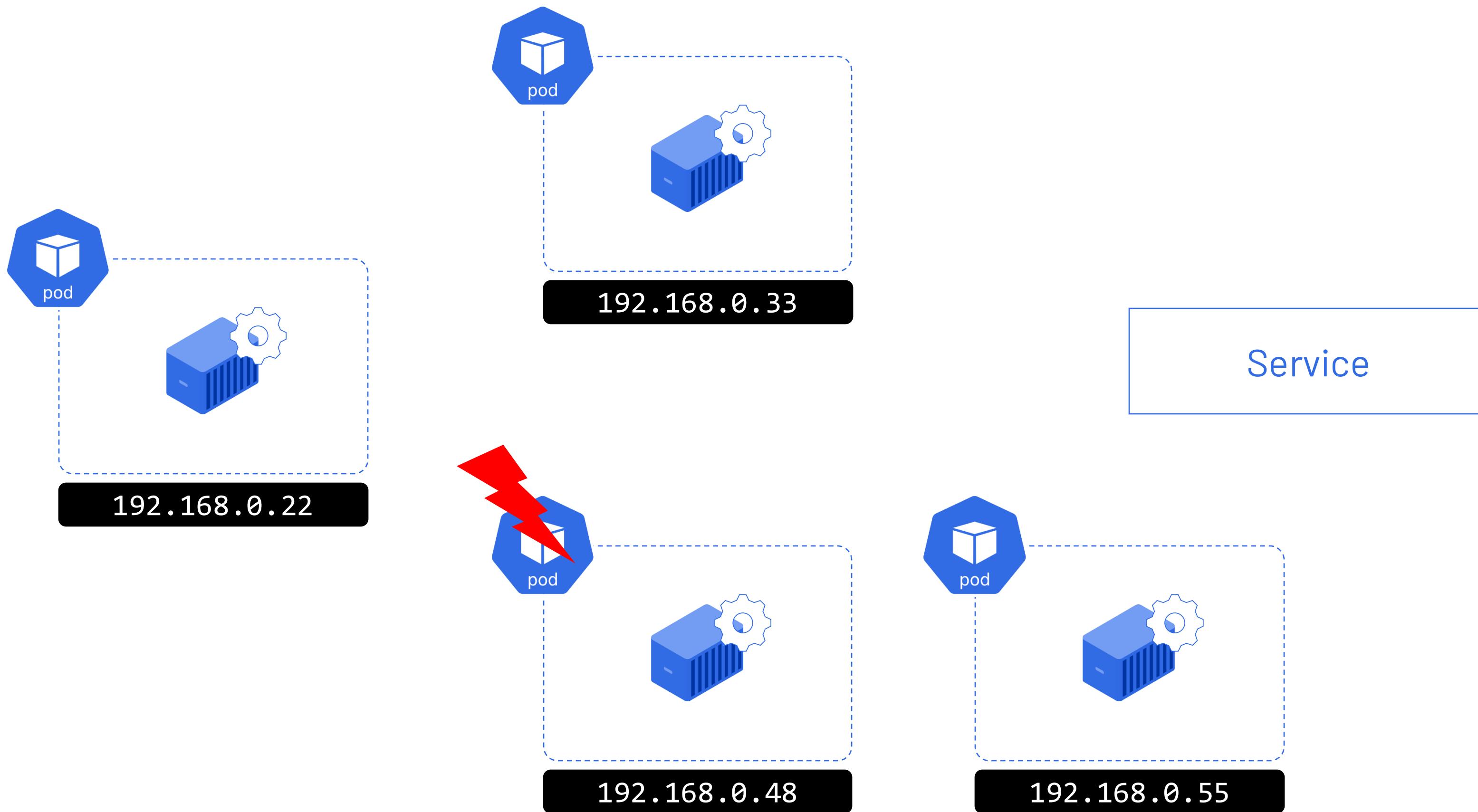
Demo

Useful kubectl
Tips

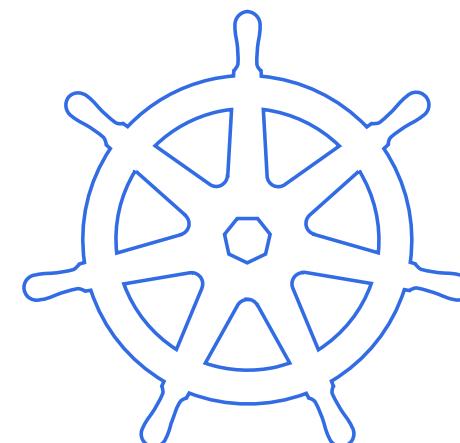
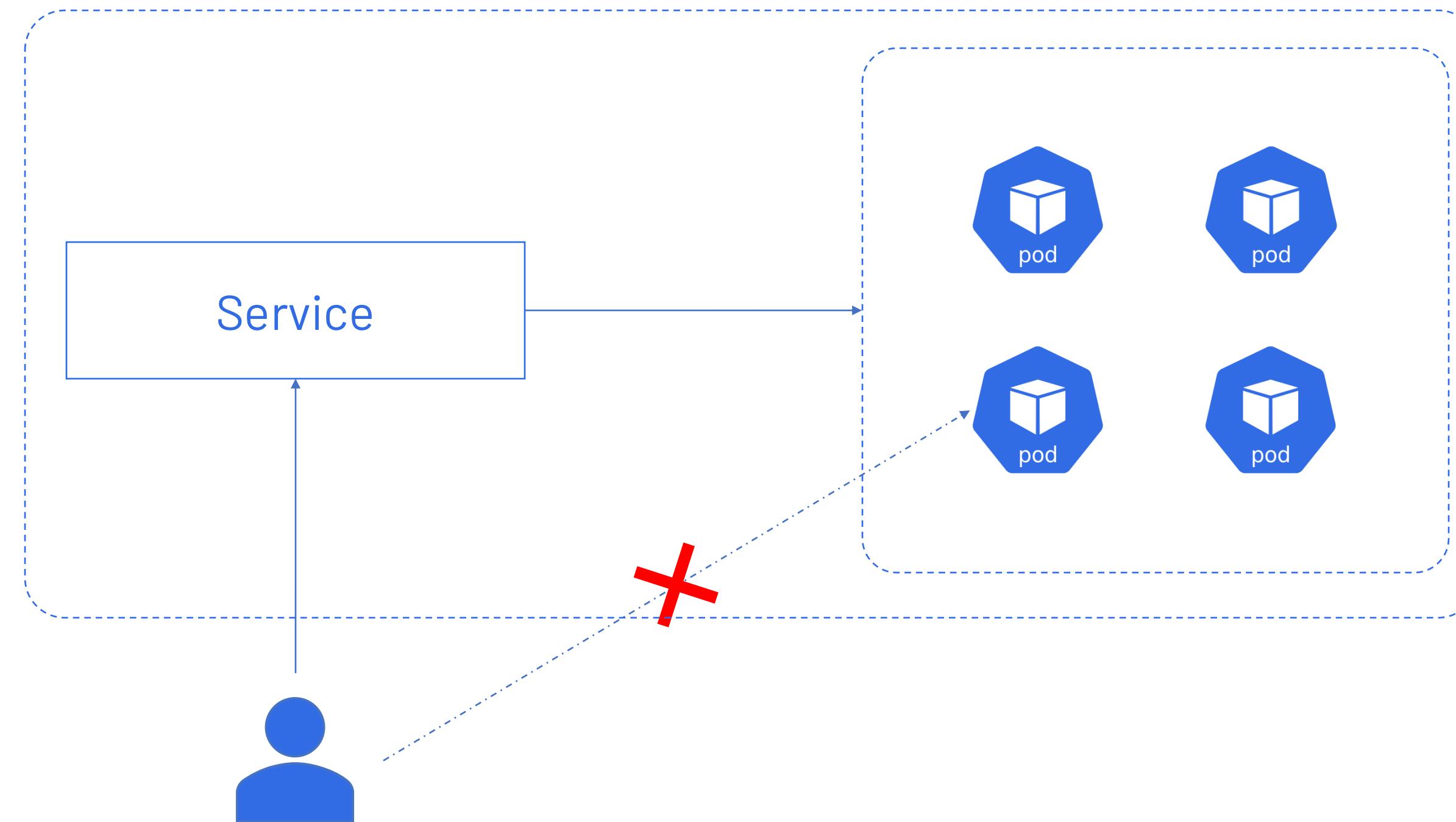


Services

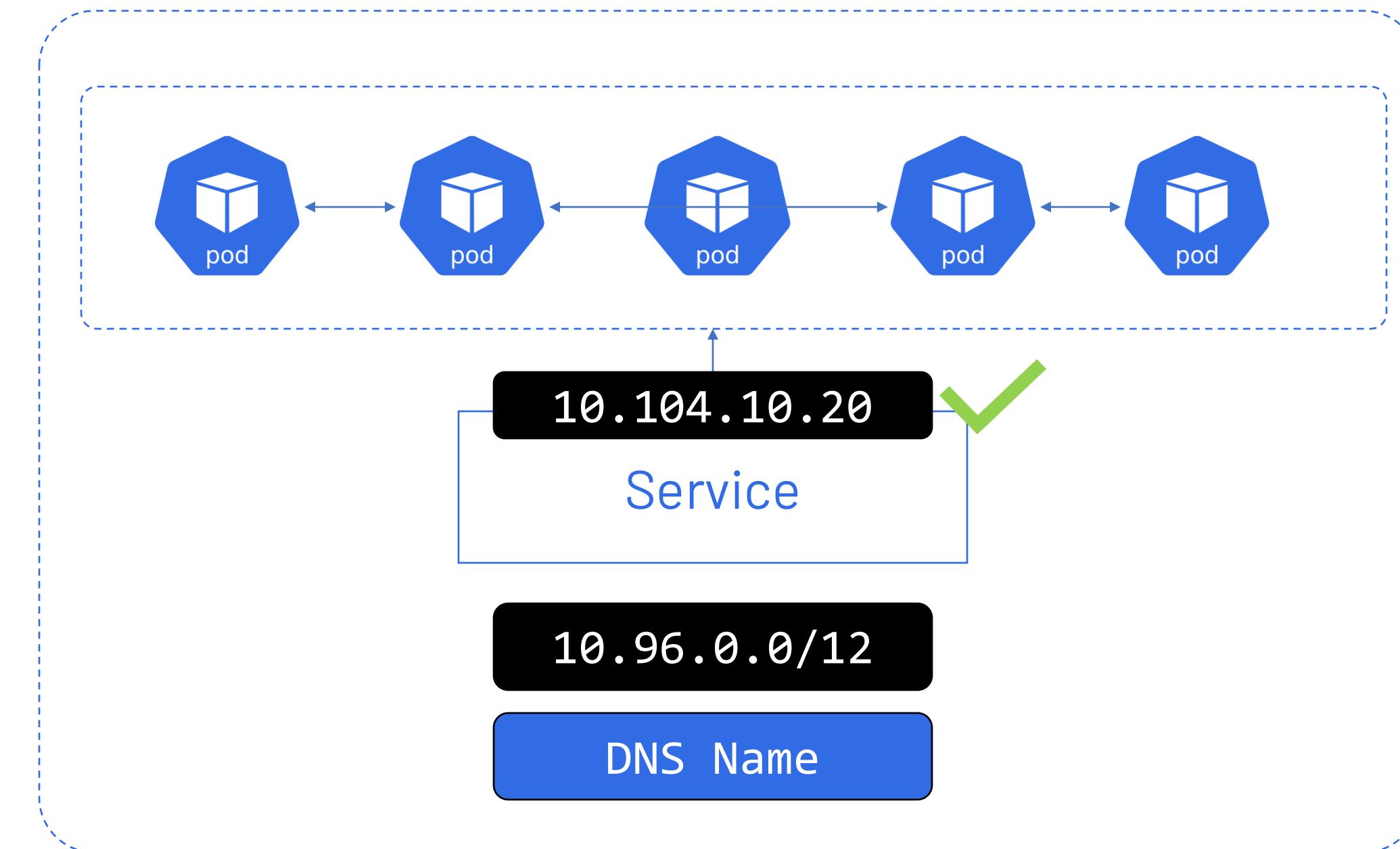
Services



Services

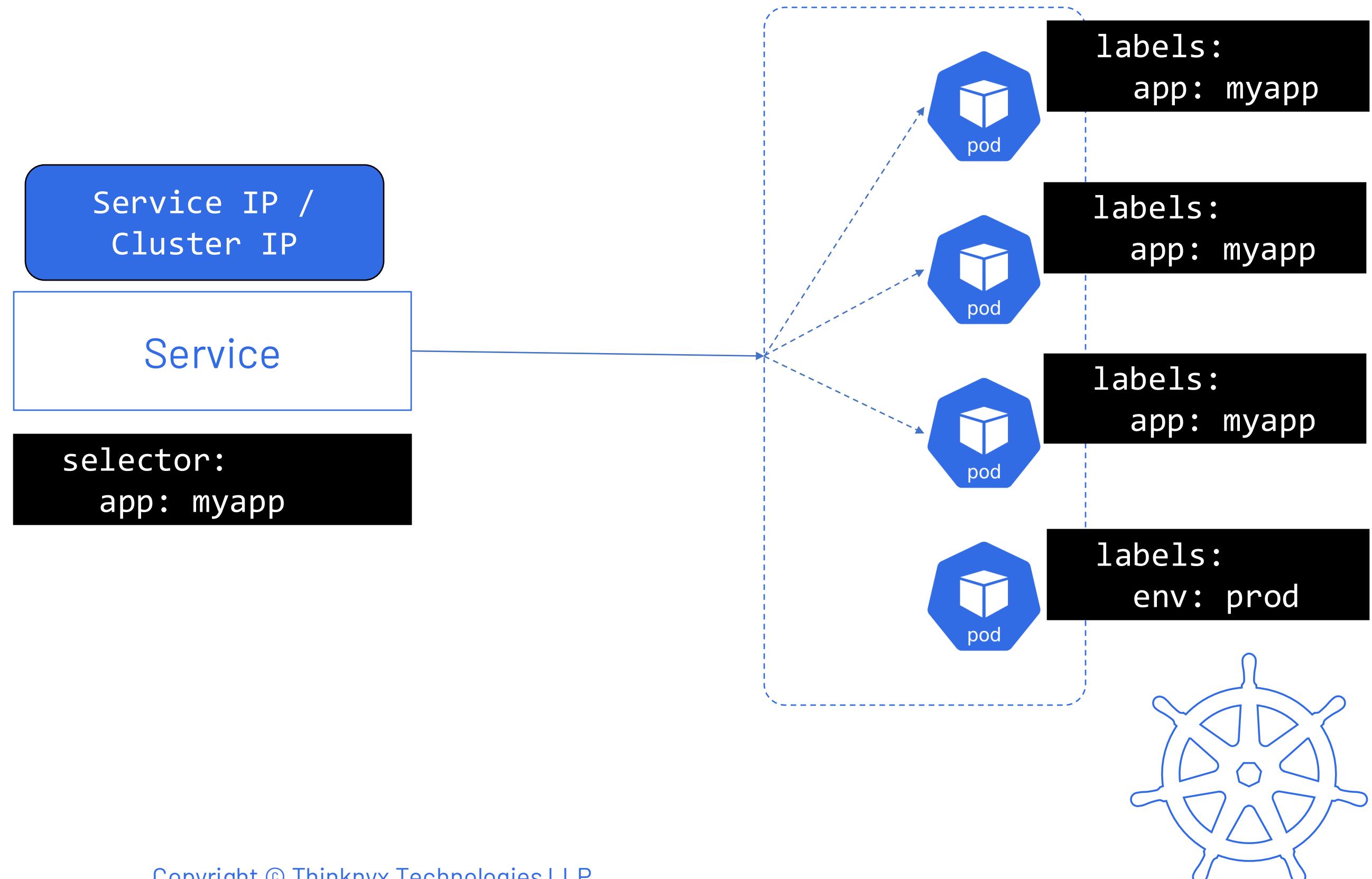


Services



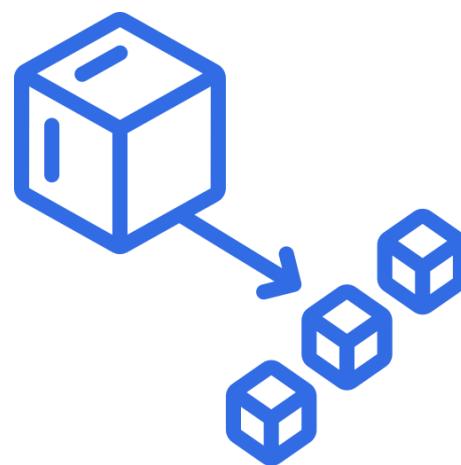
Services

Services can
perform Internal
load balancing

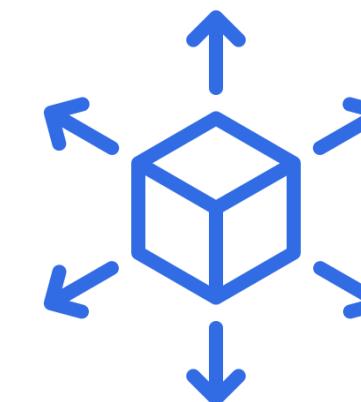


Benefits of Service Objects

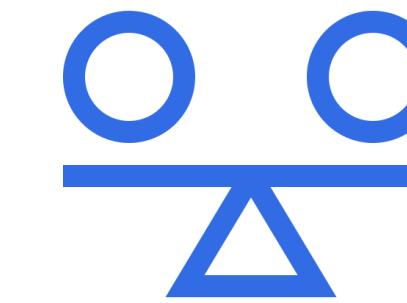
Benefits of Service Objects



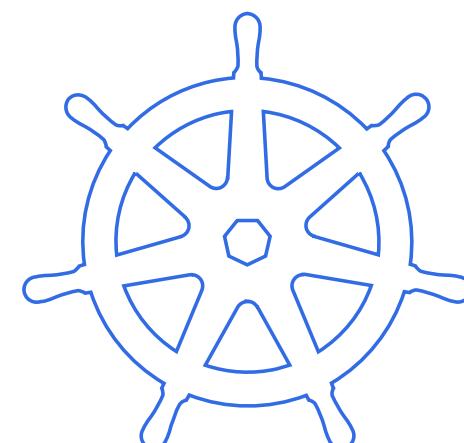
Decoupling



Scalability



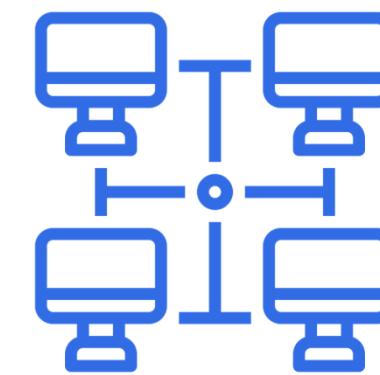
Stability



ClusterIP

ClusterIP

- Expose applications exclusively to other pods within the same cluster for internal communication



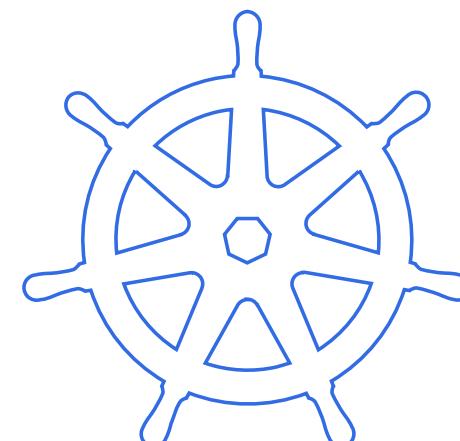
Communication



Security



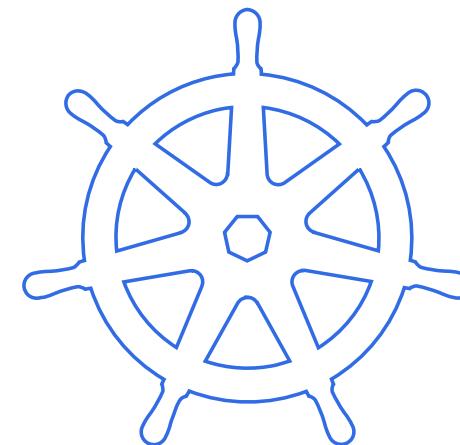
Limitations





Demo

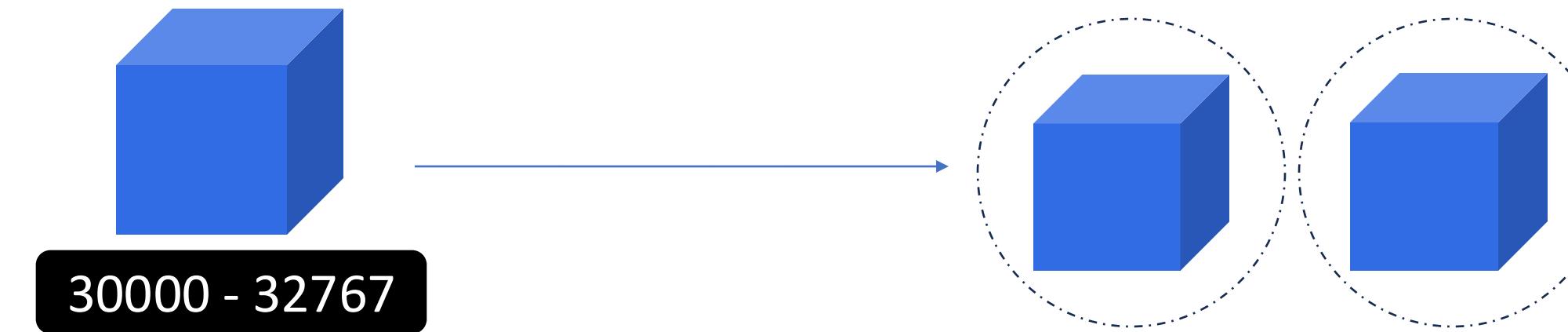
ClusterIP
Service Type



NodePort

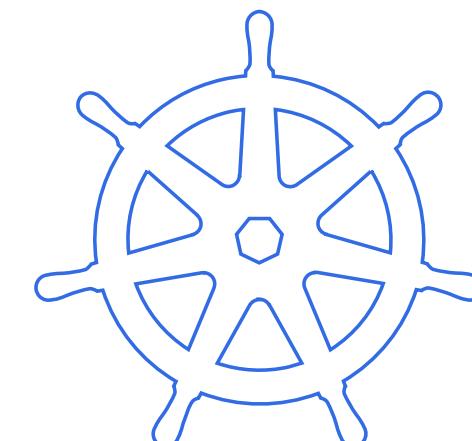
NodePort

- Expose application on a static port across all nodes in the cluster
- Default port range is 30000 – 32767



NodePort Benefits

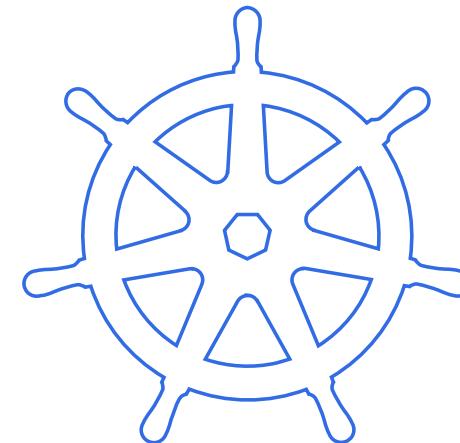
- ✓ Simple External Access
- ✓ No Load Balancer Required





Demo

NodePort
Service Type



LoadBalancer

LoadBalancer

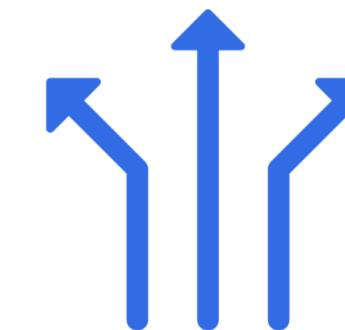
- Leverages external or cloud provider-specific load balancers to distribute incoming traffic across pods
- Highly available & scalable solution for exposing application to the public internet



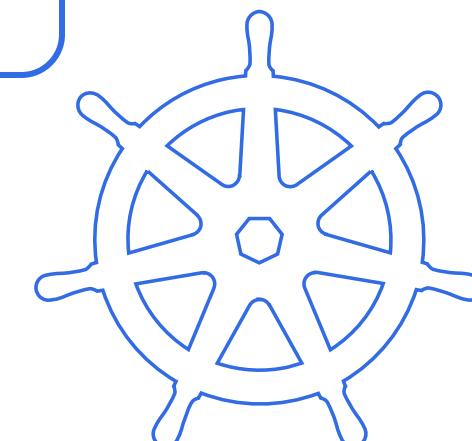
Cloud Provider
Integration



External IP
Address



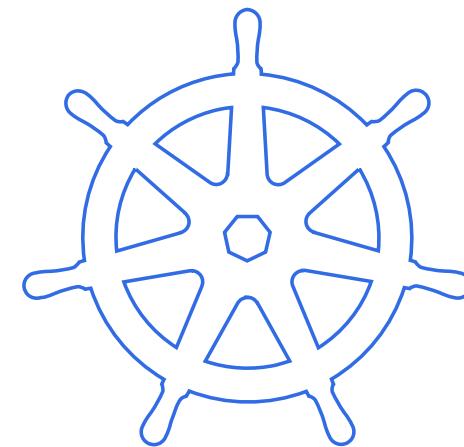
Traffic
Distribution





Demo

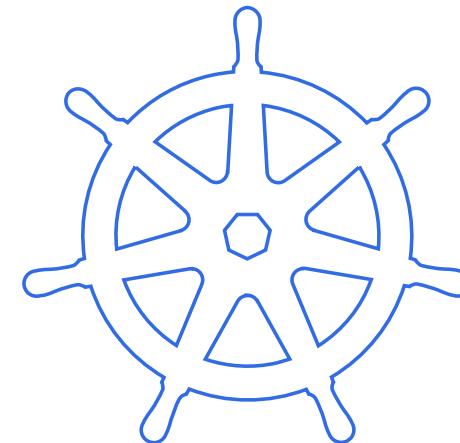
LoadBalancer
Service Type





Demo

Troubleshooting
Deployment and Service

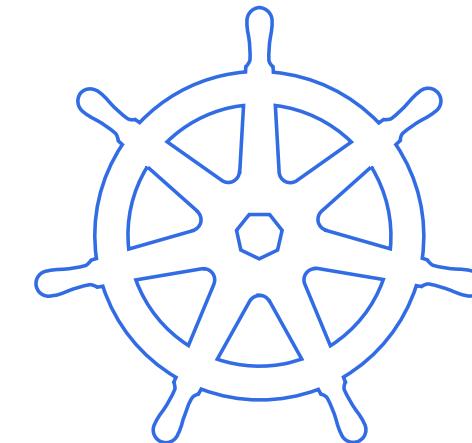


Understanding Kubernetes API Ecosystem



Demo

Kubernetes API ecosystem &
developing object definition
files

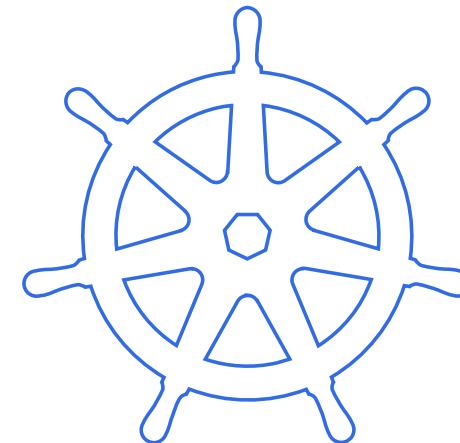


YAML Basics



Demo

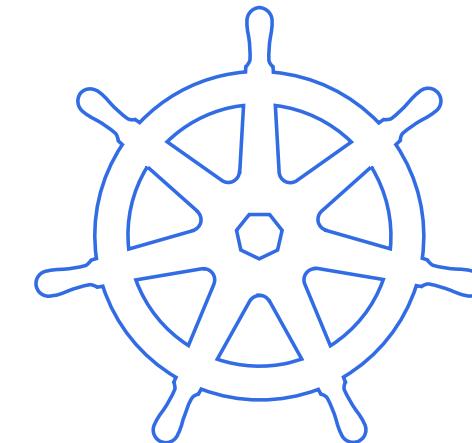
Deployment & Service
Creation





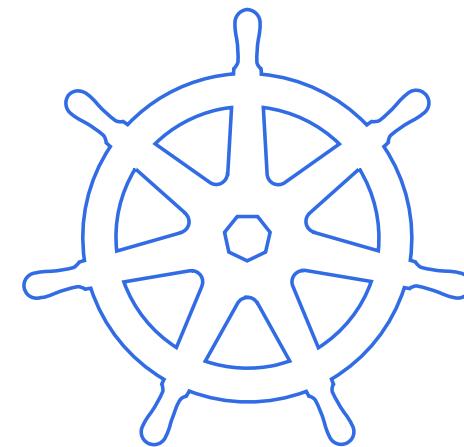
Demo

Smart way to create
Kubernetes objects





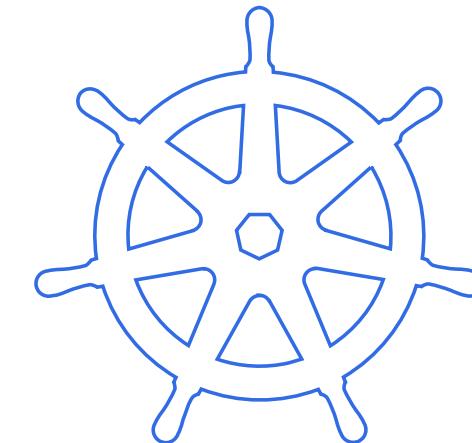
Demo | kubectl





Demo

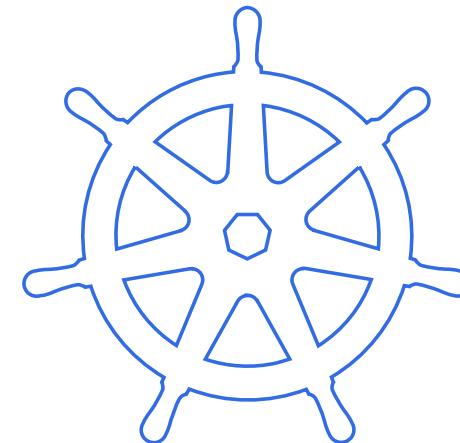
Deploy the Python
app on Kubernetes





Demo

Scale the
application

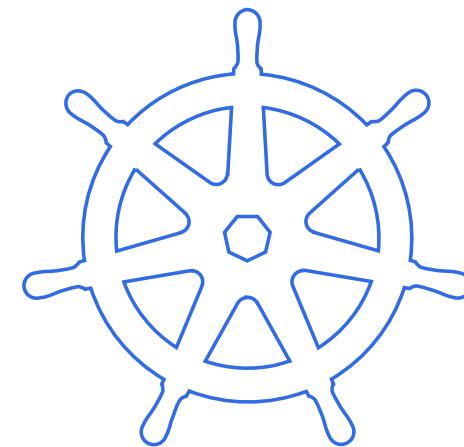


Summary



Demo

Updating k8s
Progress in
GitHub Projects



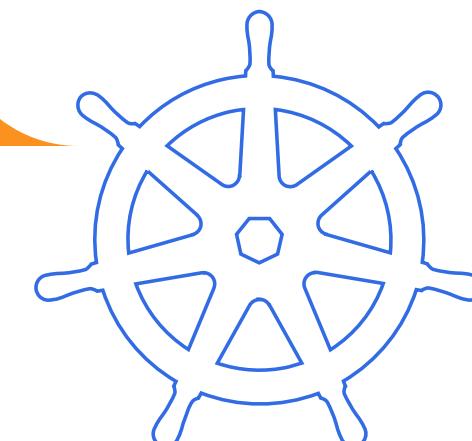
Summary

Limitations of containers and need for orchestration

Introduction to Kubernetes, purpose, architecture, setup options

Two-node cluster setup with Kubeadm

Exploring Kubernetes CLI (kubectl)



Summary

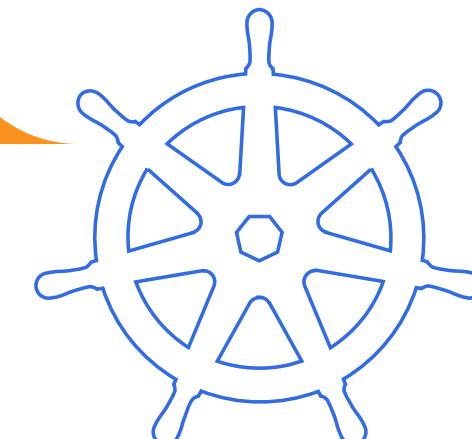


Core objects: Namespaces, Pods,
ReplicaSets, Deployments

Labels & Selectors for organizing and
managing resources

Useful kubectl tips

Service object and types: ClusterIP,
NodePort, LoadBalancer



Summary

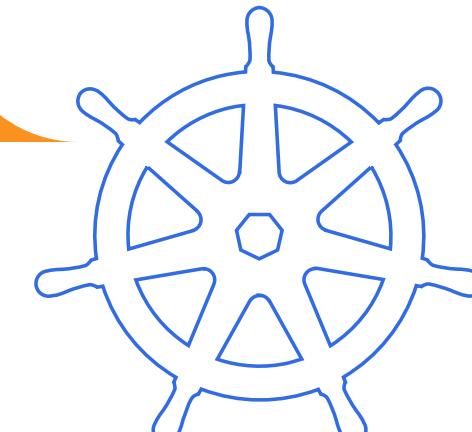


Kubernetes API ecosystem overview

Object definition files and YAML best practices

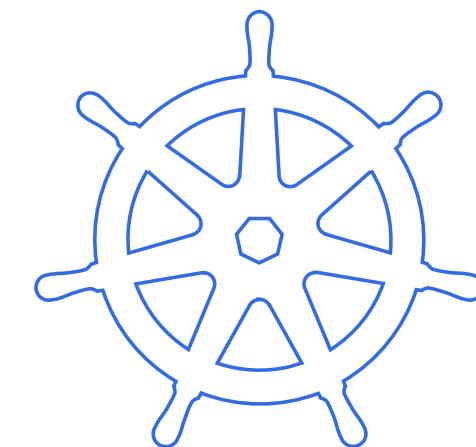
Hands-on project: Deploy and scale Python application on Kubernetes

End-to-end real-world application management demonstration





Practical Kubernetes: Beyond CKA and CKAD | Hands-on



Section 9

Monitoring with Prometheus

Presented By: Thinknyx

Getting started with Prometheus

Getting started with Prometheus

System monitoring and alerting toolkit



Open-
Source



2016

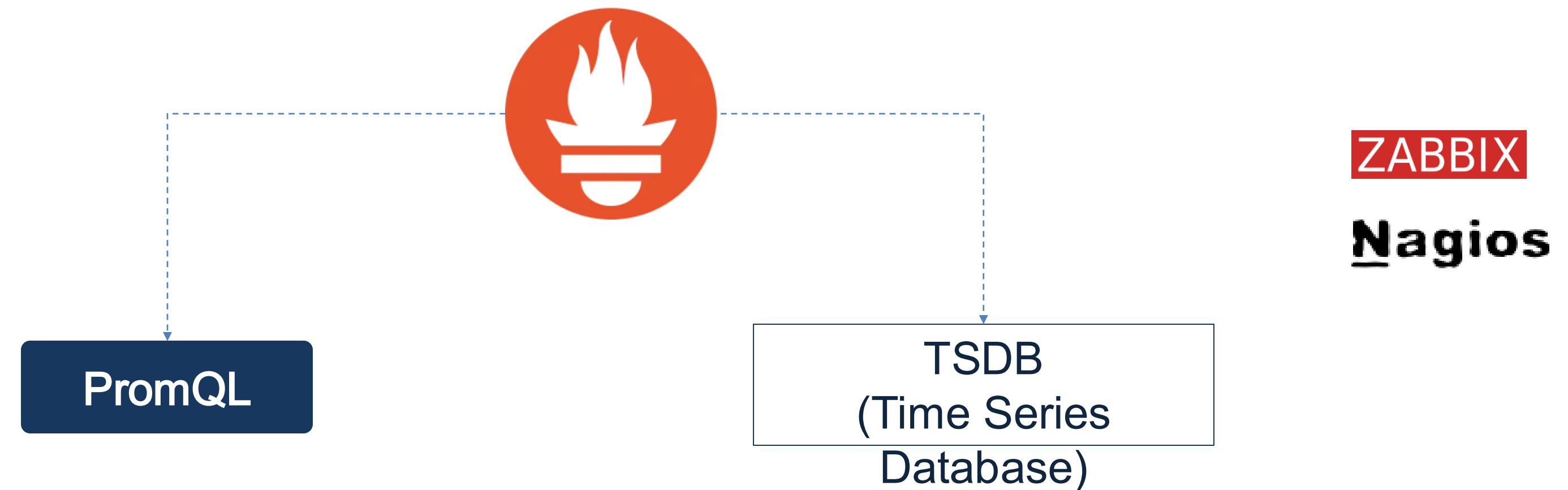
2018



Graduated
CNCF
project

Getting started with Prometheus

- Platform-agnostic and can be installed on various operating systems and environments
- Capability to monitor wide range of technologies, including databases, hardware, messaging systems, & Kubernetes



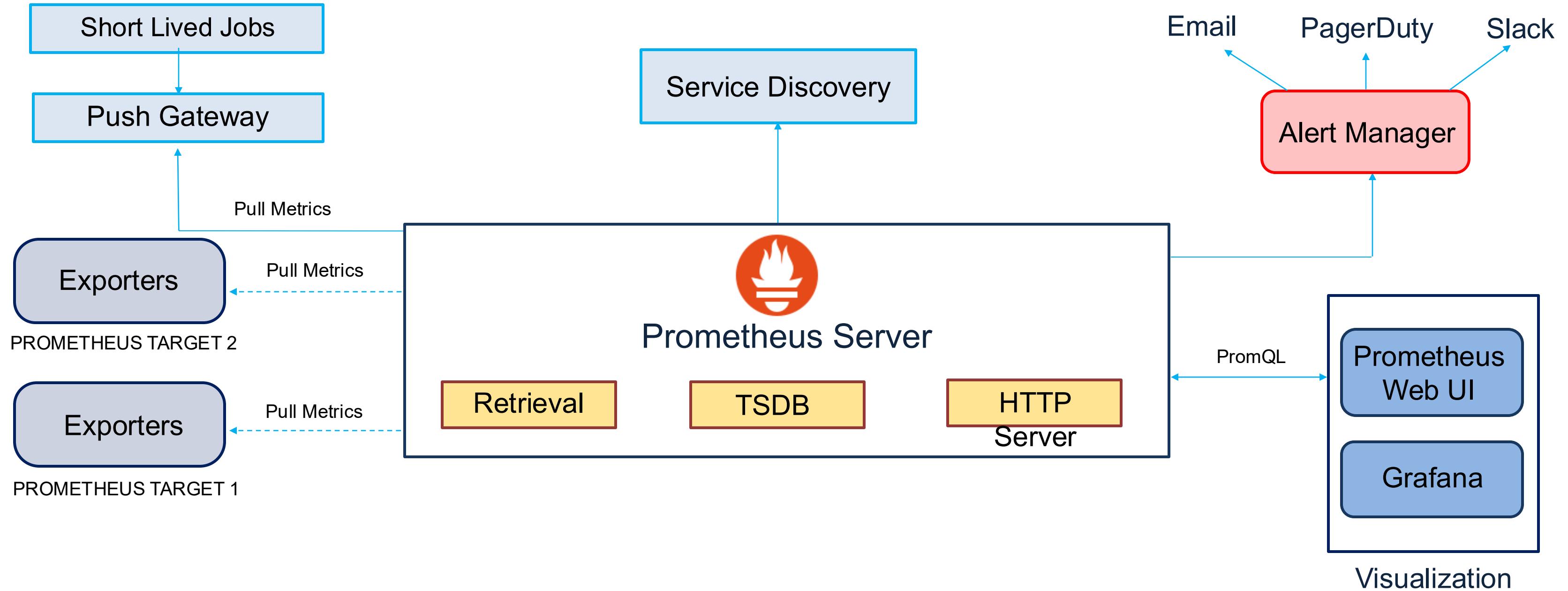
Getting started with Prometheus

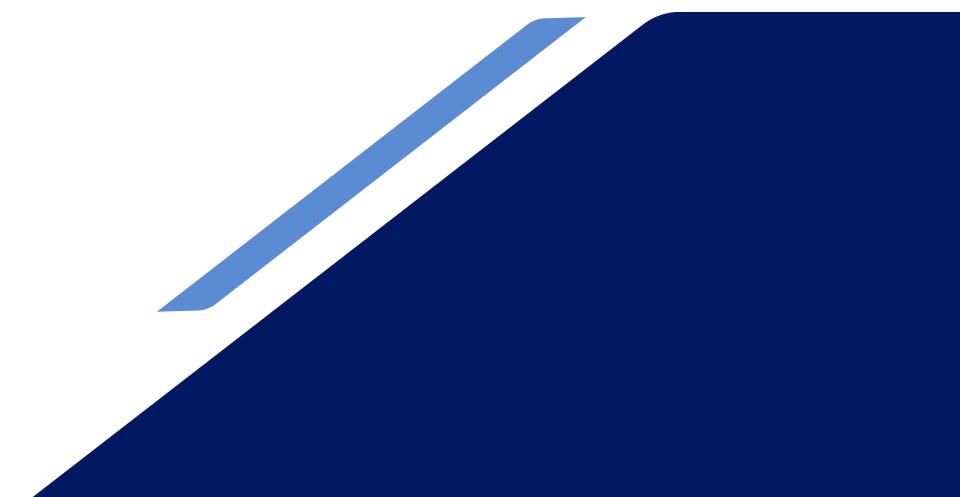
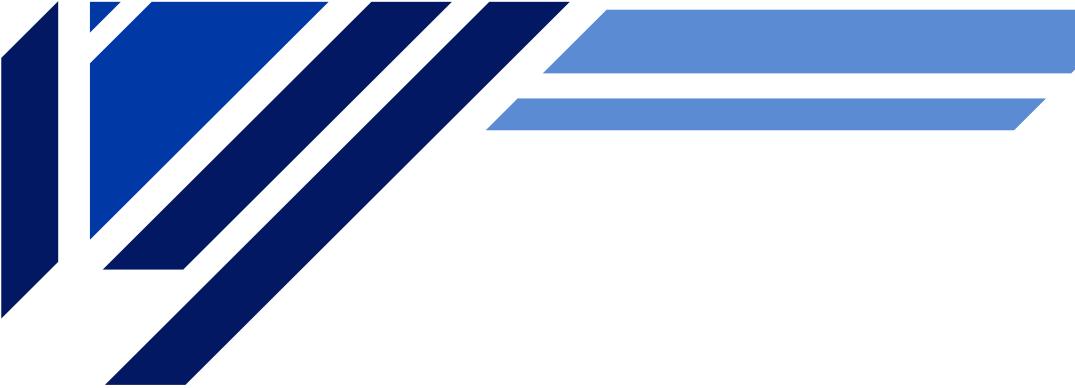


Prometheus is a monitoring system that is completely open source, CNCF-approved, platform-agnostic, and comes with the powerful PromQL query language and uses time series database (TSDB) for storing metrics

Prometheus Architecture

Prometheus Architecture





Prometheus Terminologies

Prometheus Terminologies

Target

Job

PromQL

Functions

Alerting rules & Alerts

Exporter

TSDB

Data Types

Recording Rules

Client Libraries



Prometheus Terminologies

Target

TSDB

Recording Rules

Exporter

PromQL

Alerting Rules



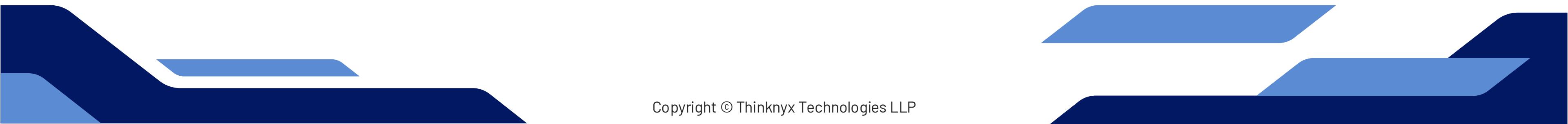
Client Libraries

Demo

DevOps | Prometheus Setup and Kubernetes Metrics



THANK YOU



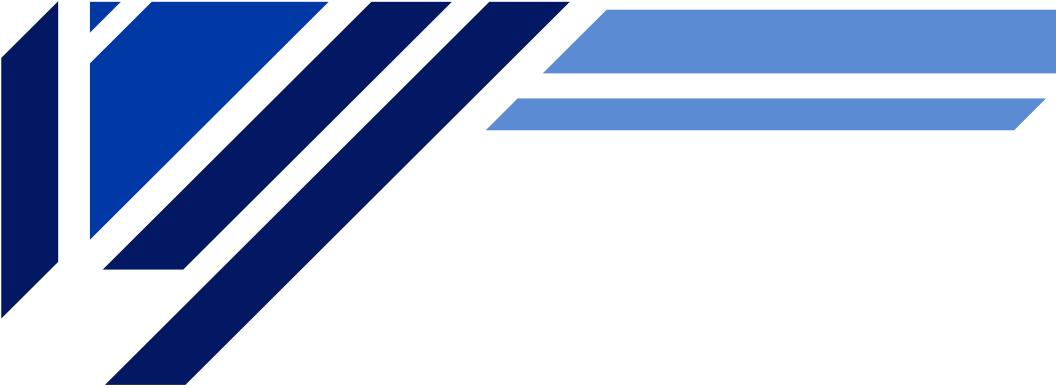
Demo

DevOps | Updating Prometheus Progress in GitHub Projects

Section 10

Visualization with Grafana

Presented By: Thinknyx



Getting started with Grafana



Getting started with Grafana



Open-
Source



User friendly
dashboard



Community-based
Dashboard



Alert
Management

Getting started with Grafana



Grafana OSS (Open
Source Software)

Grafana Enterprise

Grafana Cloud

Getting started with Grafana



**Grafana
as a solution**

Data Analytics

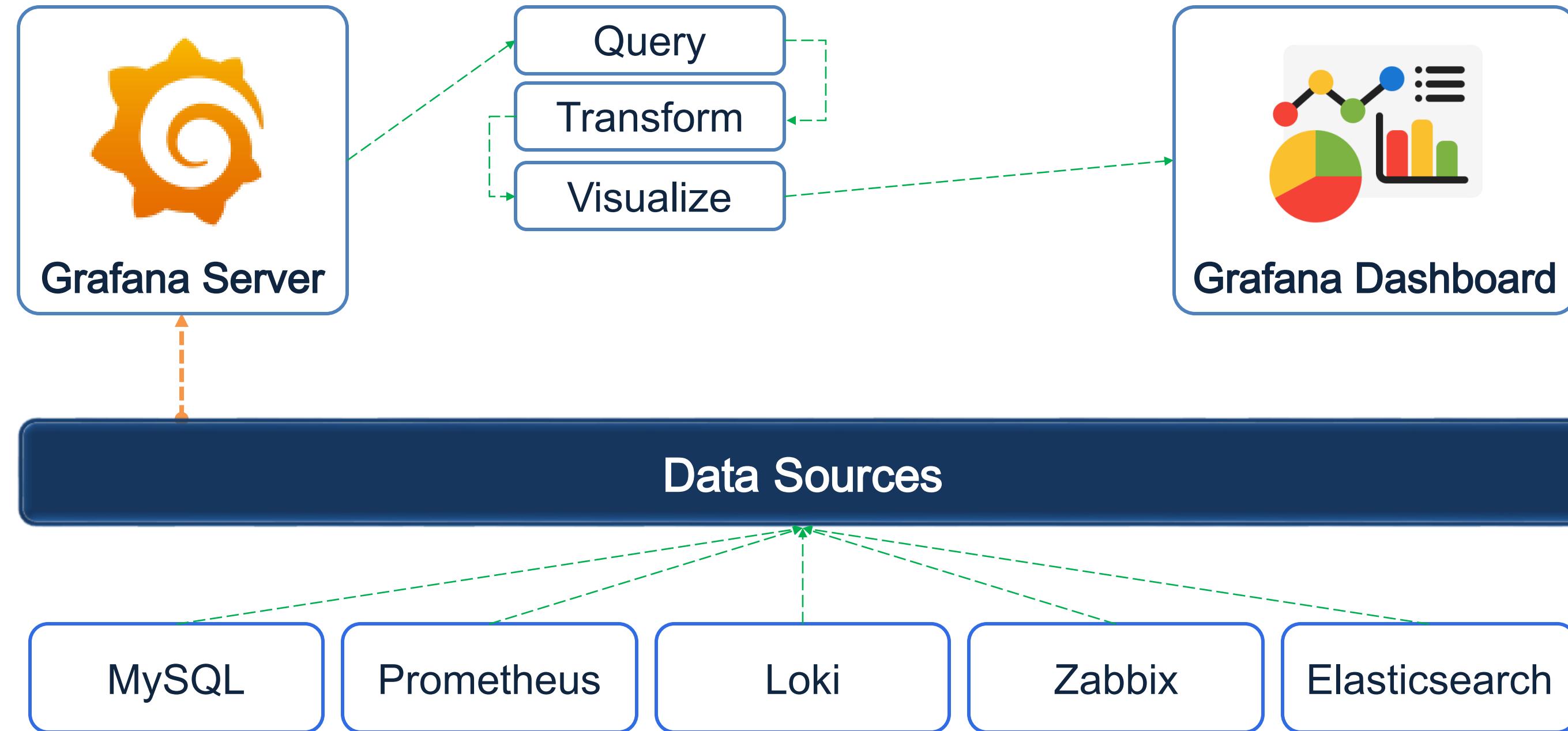
Querying and visualizing metrics

Custom dashboards

Achieving crystal-clear observability

Grafana Architecture

Grafana Architecture





Grafana Terminologies



Grafana Terminologies



Data Sources

- Data sources are the foundation of Grafana
- Define where data comes from
- **Examples:** Prometheus, InfluxDB, Loki, Elasticsearch, MySQL, etc.
- Grafana connects to data sources for data collection
- Enables querying and visualization



Dashboards

- Dashboard = collection of panels in a single view
- Enables monitoring and analysis of multiple metrics
- Provides centralized visibility in one place
- Tailored for specific use cases:
- Infrastructure health
- Application performance
- Business metrics



Visualizations

- Visualizations represent data in panels
- Grafana supports multiple visualization types:
 - Line and bar graphs
 - Heatmaps
 - Gauges and pie charts
 - Tables
 - Logs and status indicators

Grafana Terminologies



Explore

- Run ad-hoc queries
- Troubleshoot issues
- Inspect data interactively (outside dashboards)
- Useful for developers and operators
- Enables deeper analysis of live data



Alert Management

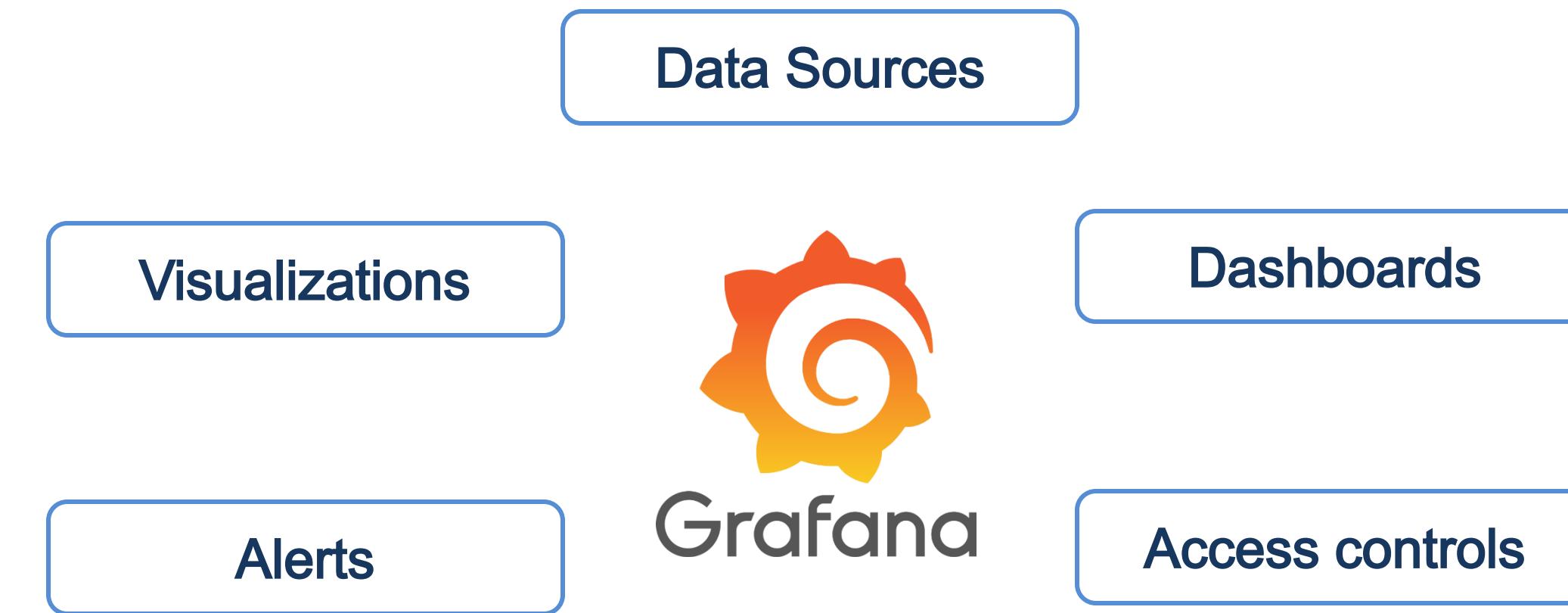
- Set up alerts based on panel queries
- Define conditions (e.g., CPU > 80%)
- Trigger alerts through channels (email, Slack, PagerDuty, etc.)
- Real-time issue notifications for teams



RBAC (Role-Based Access Control)

- RBAC manages user permissions in Grafana
- Controls who can view, edit, or administer
- Applies to dashboards, data sources, and settings
- Ensures data security in team environments
- Provides proper access control

Grafana Terminologies



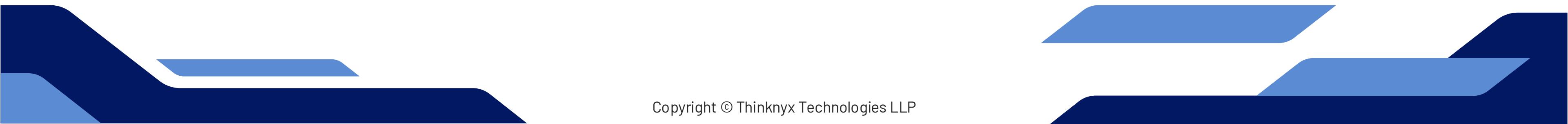
- Each component enhances observability, monitoring, and decision-making

Demo

DevOps | Grafana Dashboards for Kubernetes



THANK YOU



Demo

DevOps | Updating Grafana Progress in GitHub Projects

Section 11

GitOps with ArgoCD

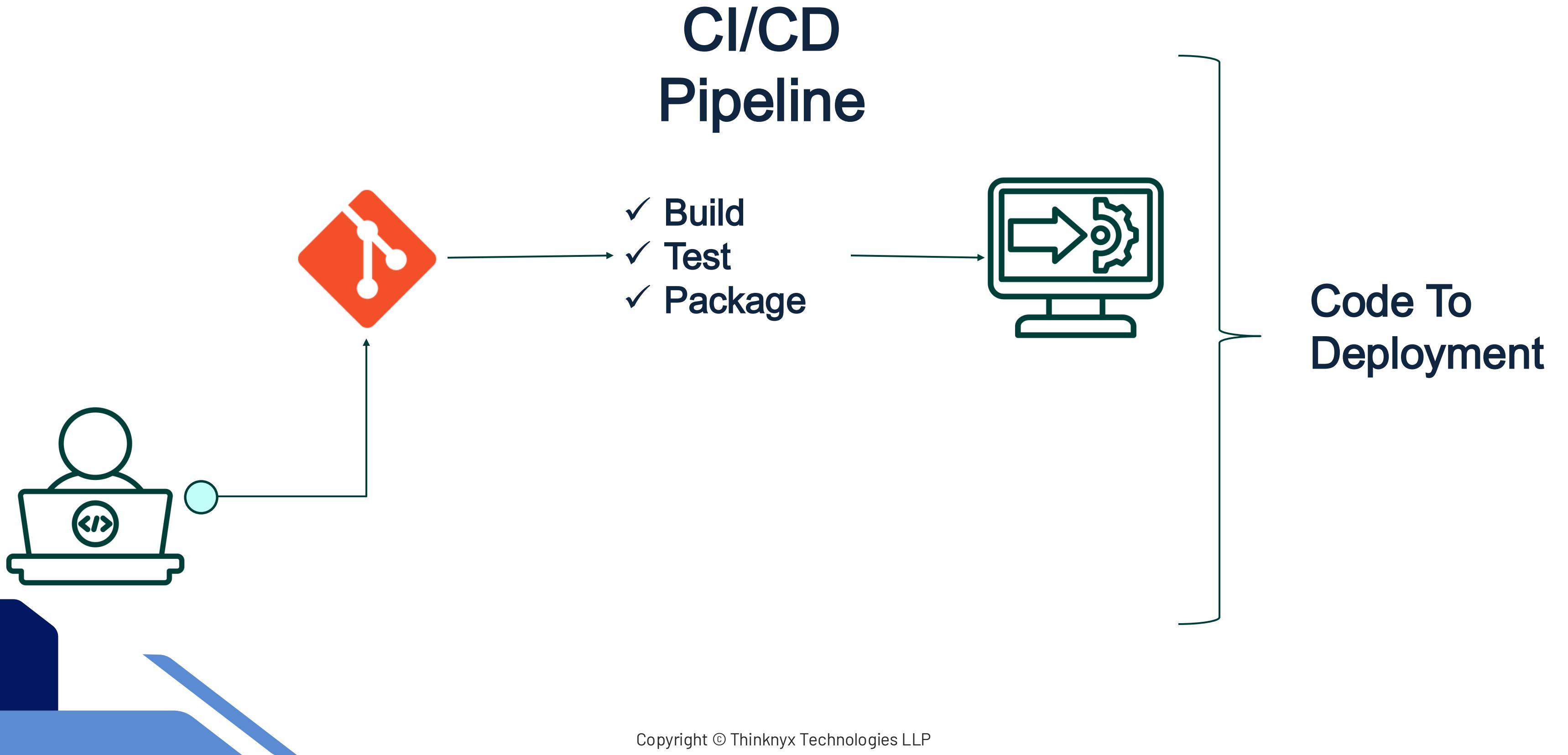
Presented By: Thinknyx

Introduction to GitOps

Introduction to GitOps

GitOps

Introduction to GitOps



Introduction to GitOps



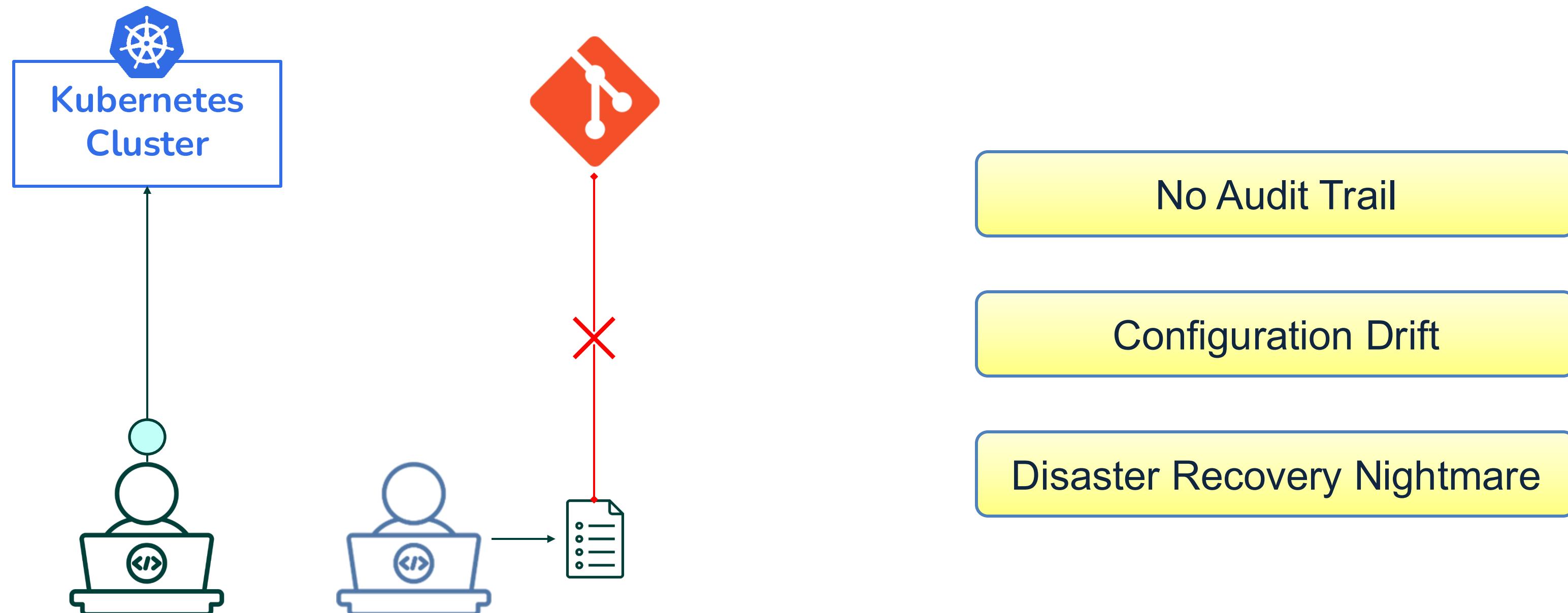
Who is managing the infrastructure configurations?

How are changes to Kubernetes manifests or cluster configurations applied?



- Manually running `kubectl` commands
- Manually editing YAML files

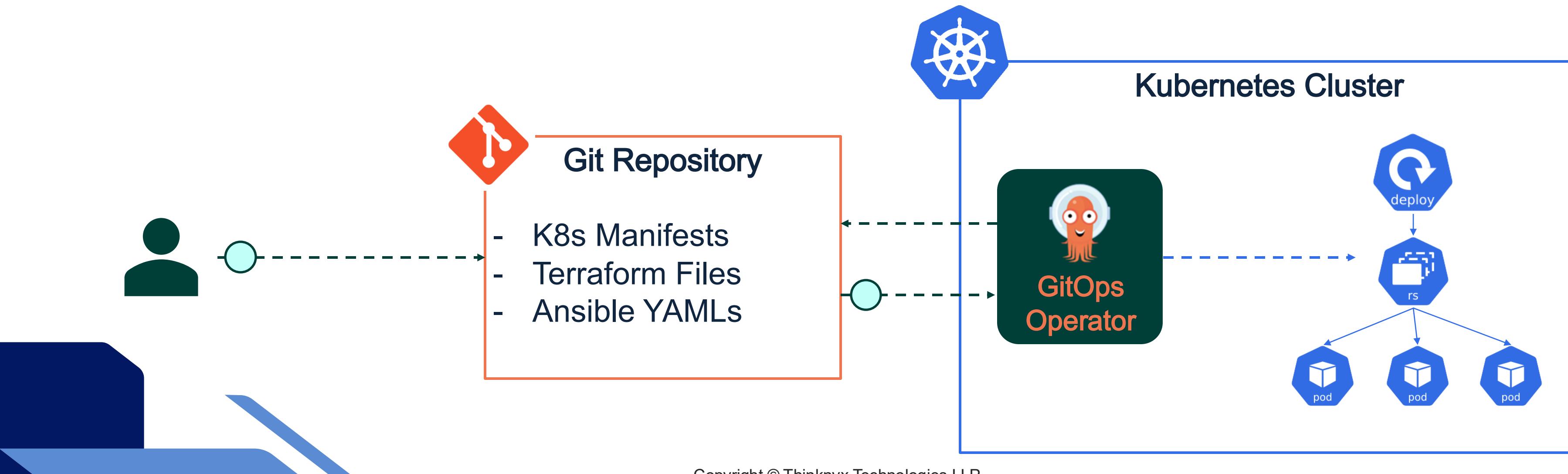
Introduction to GitOps



Introduction to GitOps

GitOps

Infrastructure & cluster configurations



Introduction to GitOps

GitOps

- ✓ Every change is audited in Git.
- ✓ Configurations are versioned and recoverable.
- ✓ The cluster always matches what's in Git — no drift, no mystery changes.

Introduction to GitOps

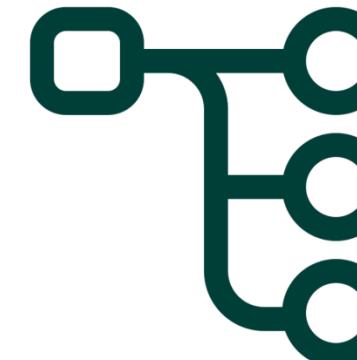
GitOps

GitOps is a modern approach to continuous deployment that uses Git as the single source of truth for your application code & infrastructure configurations

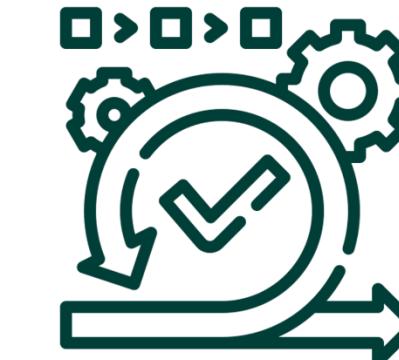
Four Principles of GitOps



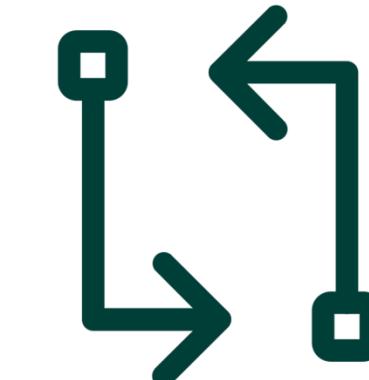
Declarative Configuration



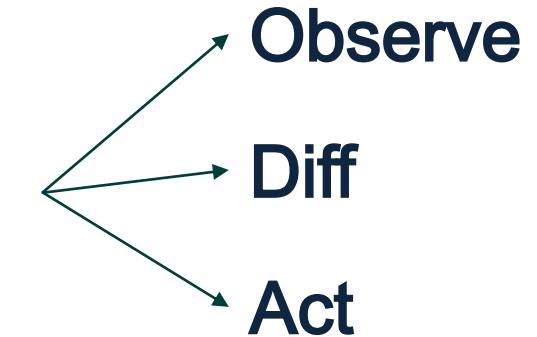
Version Control



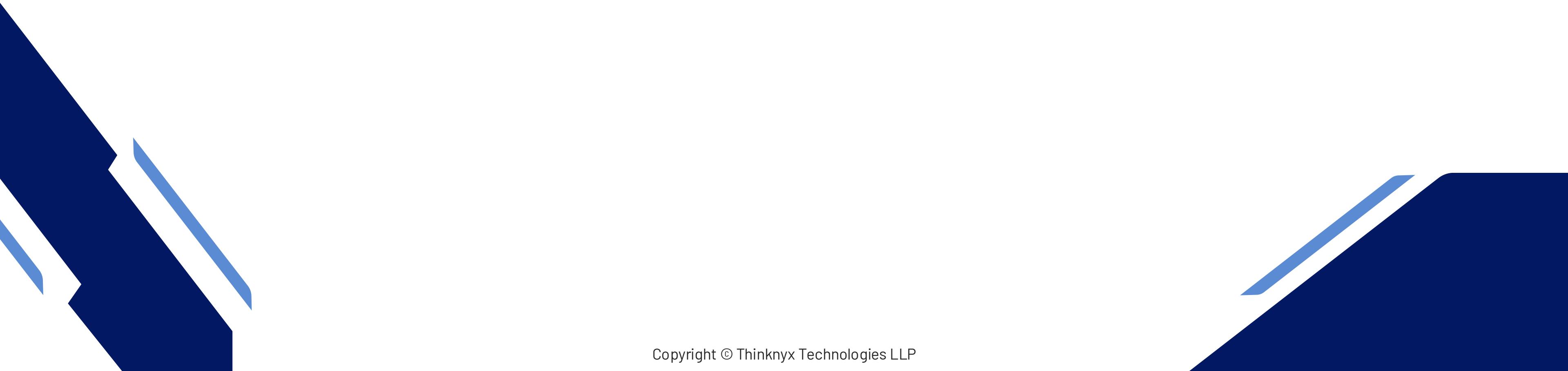
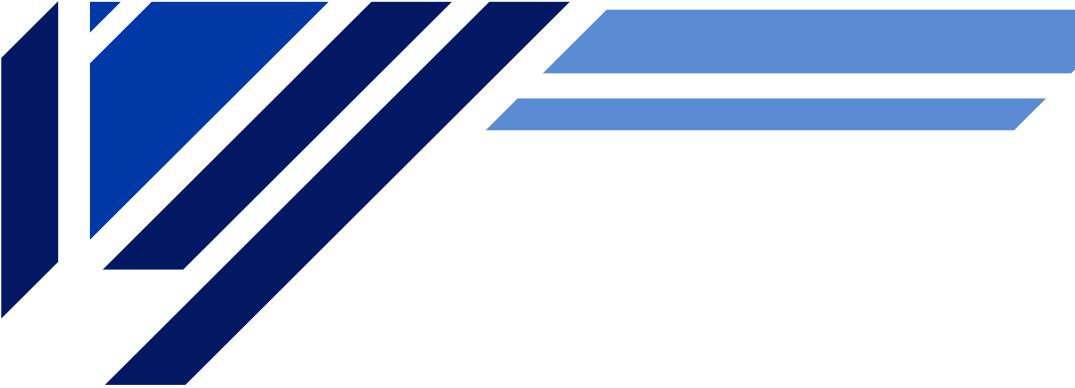
Automated Delivery



Continuous Reconciliation

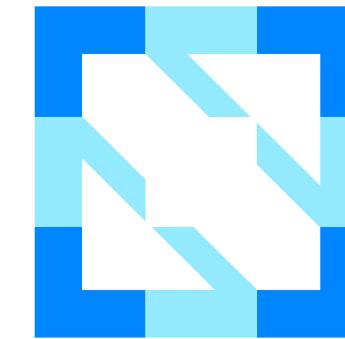


Observe
Diff
Act



Overview of the Argo Project's

Argo Projects



**CLOUD NATIVE
COMPUTING FOUNDATION**

Argo Projects



Argo
Workflows

Workflow engine for orchestrating tasks
like CI pipelines, data processing, and
batch jobs

Argo Projects



Argo CD

GitOps-based continuous delivery tool, ensures Kubernetes clusters are synchronized with Git repositories, automating deployment and infrastructure management

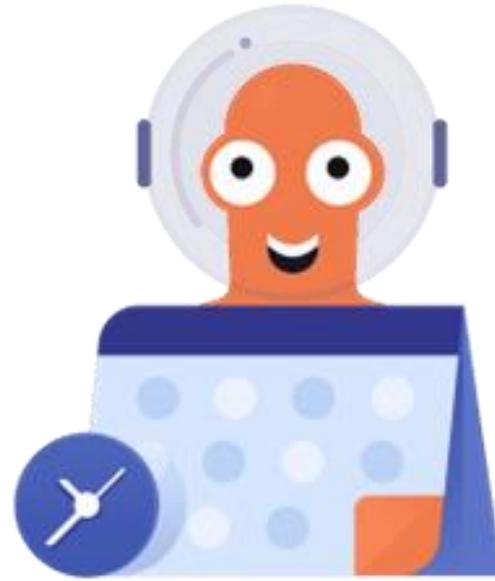
Argo Projects



Argo Rollouts

Controller for advanced deployment strategies, including **Canary** releases, **Blue-Green** deployments, and **Progressive Rollouts**

Argo Projects



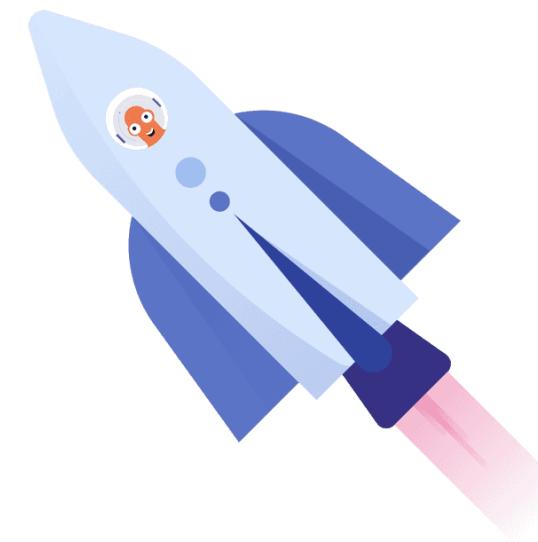
Event-driven automation framework that triggers workflows or actions based on external events

Argo Events

Argo Projects



Argo Workflows



Argo CD



Argo Rollouts



Argo Events

- ✓ **Workflows**
- ✓ **Deployments**
- ✓ **Event-driven automation**

Introduction Argo CD

What is Argo CD?

Argo CD is a declarative, GitOps-based CD tool for Kubernetes that automates app deployments. It syncs the Git-defined desired state with the actual state of the cluster.

- ✓ Monitors the **current state** of applications running in the cluster
- ✓ Detects any **deviations** from the desired state stored in Git
- ✓ Provides **visualizations** and **automated reconciliation**

Why Argo CD?

Declarative
& Version
Controlled

Automation
&
Remediation

Scalability
&
Flexibility

How Argo CD Works?



Single source of truth

How Argo CD Works?



Syncing
Applications



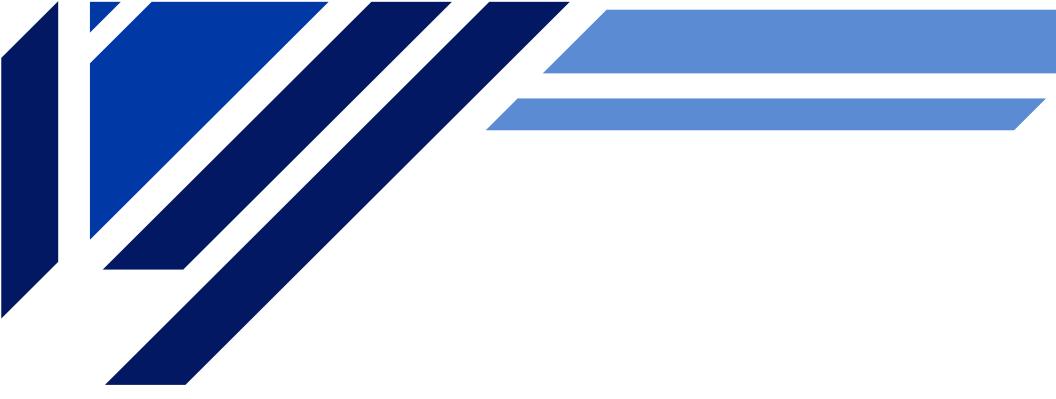
Monitoring
State



Reconciliation
Loop

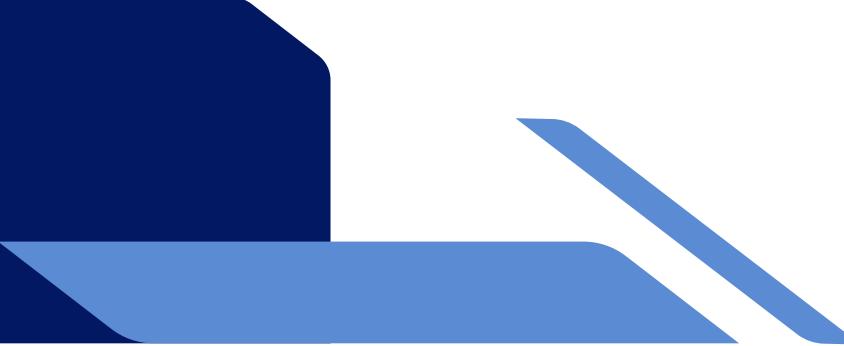


Visual
Interface



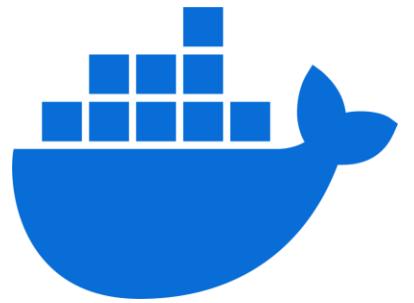
Features of Argo CD

Features of Argo CD

- 
- Automated Application Deployment
 - Support for Multiple Configuration Tools
 - Multi-Tenancy & RBAC
 - Rollback & Roll-anywhere
 - Web UI and CLI
 - Audit Trails
 - Multi-Cluster Management
 - Single Sign-On (SSO) Integration
 - Drift Detection & Synchronization
 - Health Status Analysis
 - Webhook & Automation Integration
 - Observability & Metrics

Core Argo CD Terminologies

Core Argo CD Terminologies

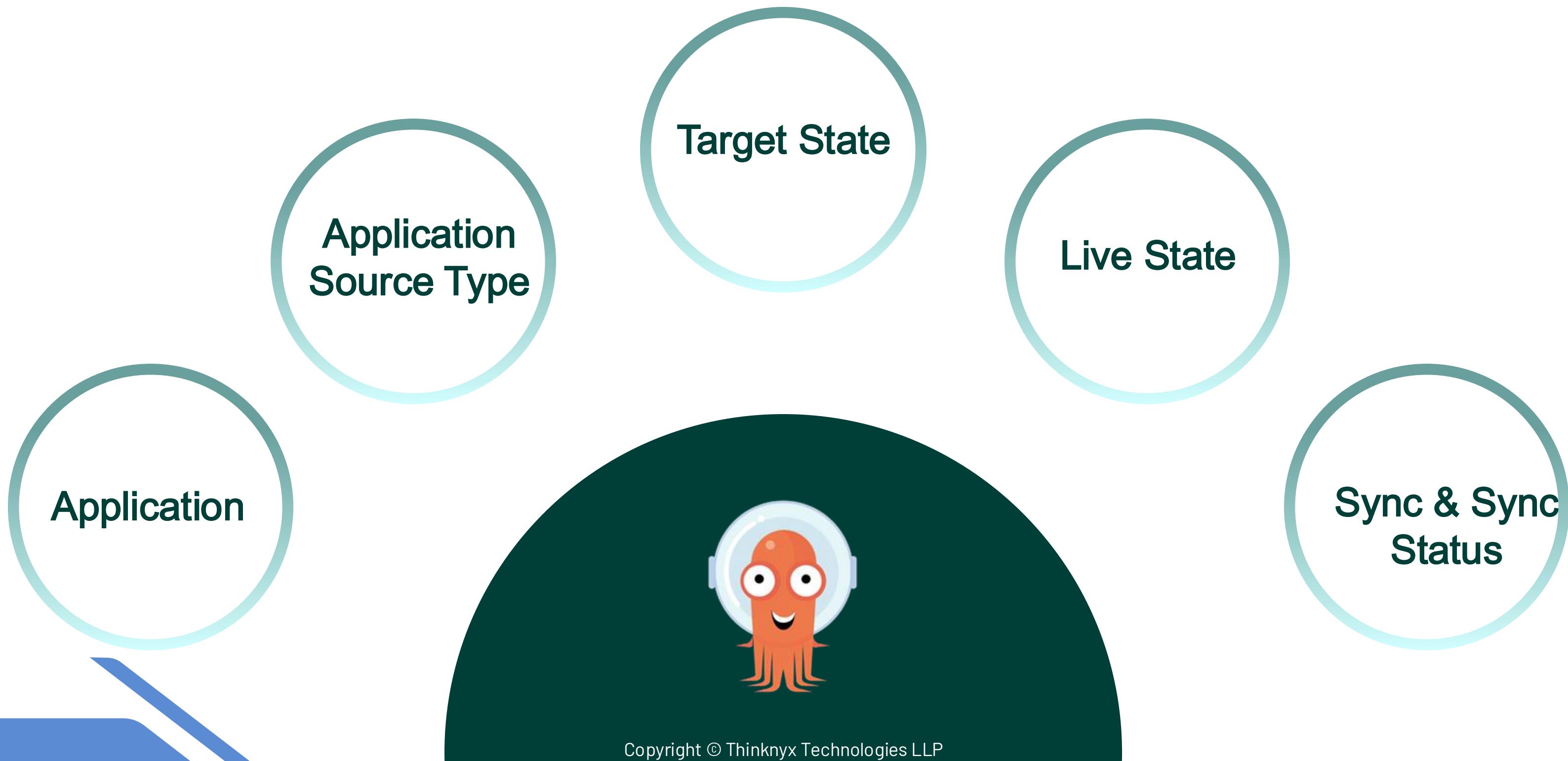


Continuous
Delivery

GitOps



Core Argo CD Terminologies





**Sync
Operation
Status**

Refresh

Health

Projects

Demo

DevOps | Install ArgoCD on Kubernetes

Demo

DevOps | Deploy application using GitOps workflow



THANK YOU

Demo

DevOps | Updating ArgoCD Progress in GitHub Projects

Section 12

Automating CI/CD with GitHub Actions

Presented By: Thinknyx

Introduction to GitHub Actions

Introduction to GitHub Actions

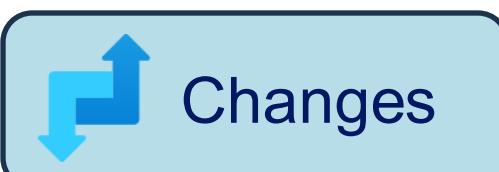
Built Python Application

Containerized into an Image

Deployed in Kubernetes
environment

Enabled Continuous Delivery
with ArgoCD

Manual Steps



Introduction to GitHub Actions

Built Python Application

Containerized into an Image

Deployed in Kubernetes environment

Enabled Continuous Delivery with ArgoCD

Manual Steps



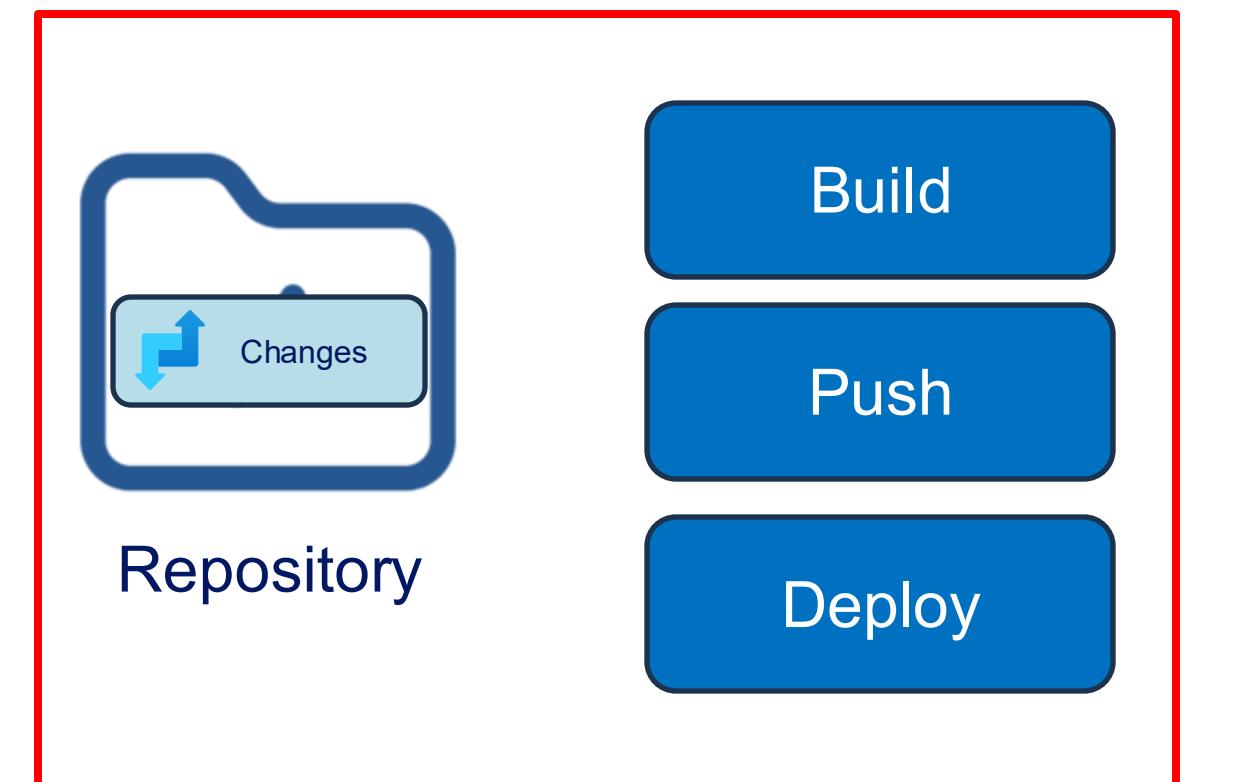
Repository

Build

Push

Deploy

Introduction to GitHub Actions

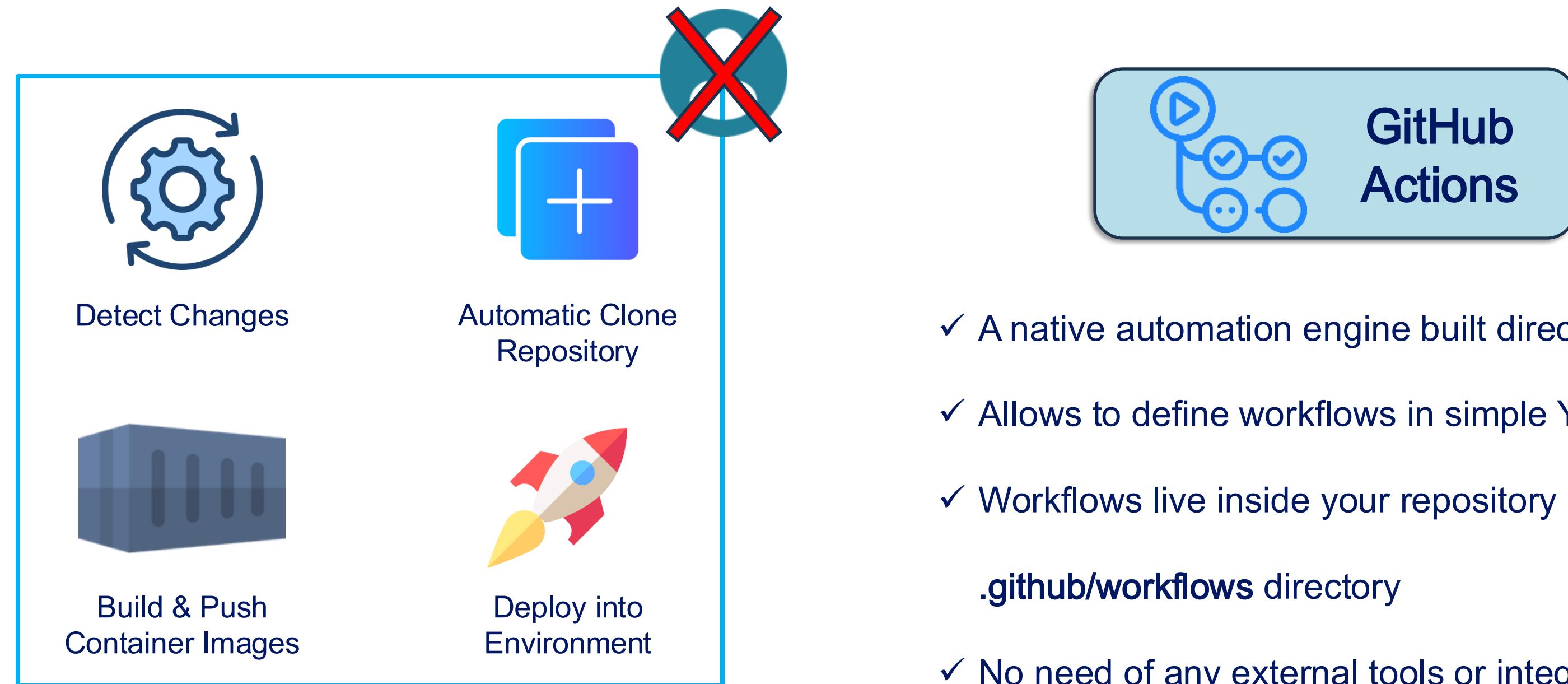


Tedious

Time-
Consuming

Error-Prone

Introduction to GitHub Actions



- ✓ A native automation engine built directly into GitHub
- ✓ Allows to define workflows in simple YAML files
- ✓ Workflows live inside your repository under the **.github/workflows** directory
- ✓ No need of any external tools or integrations
- ✓ Everything runs natively inside GitHub

Introduction to GitHub Actions

➤ Benefits



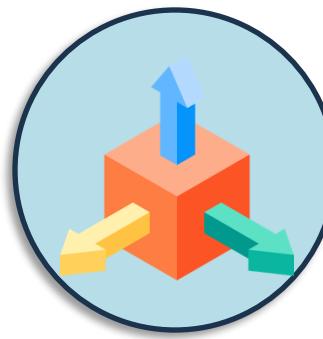
1. Built into GitHub



2. Super Easy Setup



3. Huge Community Marketplace



4. Scalability

Introduction to GitHub Actions

➤ Use Cases

Continuous Integration & Continuous Delivery / Deployment

- ✓ Auto compile, test, and feedback on code push
- ✓ Auto package/deploy on merge (with optional approval)
- ✓ Faster feedback, fewer steps, reliable deployments

Automation

- ✓ Automates small important tasks which are time-consuming
- ✓ Examples: auto-label PRs, update issues, send notifications, clean branches
- ✓ Cleaner workflow and more productive teams

DevOps Pipelines

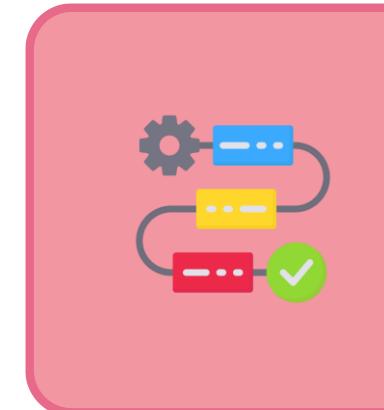
- ✓ Build complete end-to-end pipelines with GitHub Actions
- ✓ Examples: build code, run scans, check quality, publish images/charts, deploy to Kubernetes

Core Concepts of GitHub Actions

Core Concepts of GitHub Actions

Workflow

- ✓ Coordinates tasks when repo events occur
- ✓ Workflows defined in YAML under `.github/workflows`
- ✓ Answers: "When something happens, what should I automate?"



Job

- ✓ A workflow contains one or more Jobs
- ✓ Each job runs tasks on a fresh VM or container (e.g., build, test)
- ✓ Jobs can run in parallel or sequentially (with dependencies)



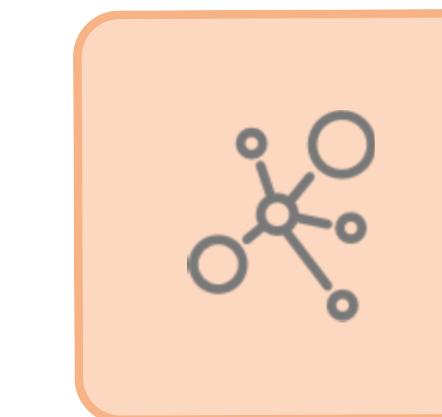
Event

- ✓ In GitHub Actions, that spark is an Event
- ✓ Events trigger workflows (push, PR, or custom activity)
- ✓ The tiny spark that powers your automation journey



Step

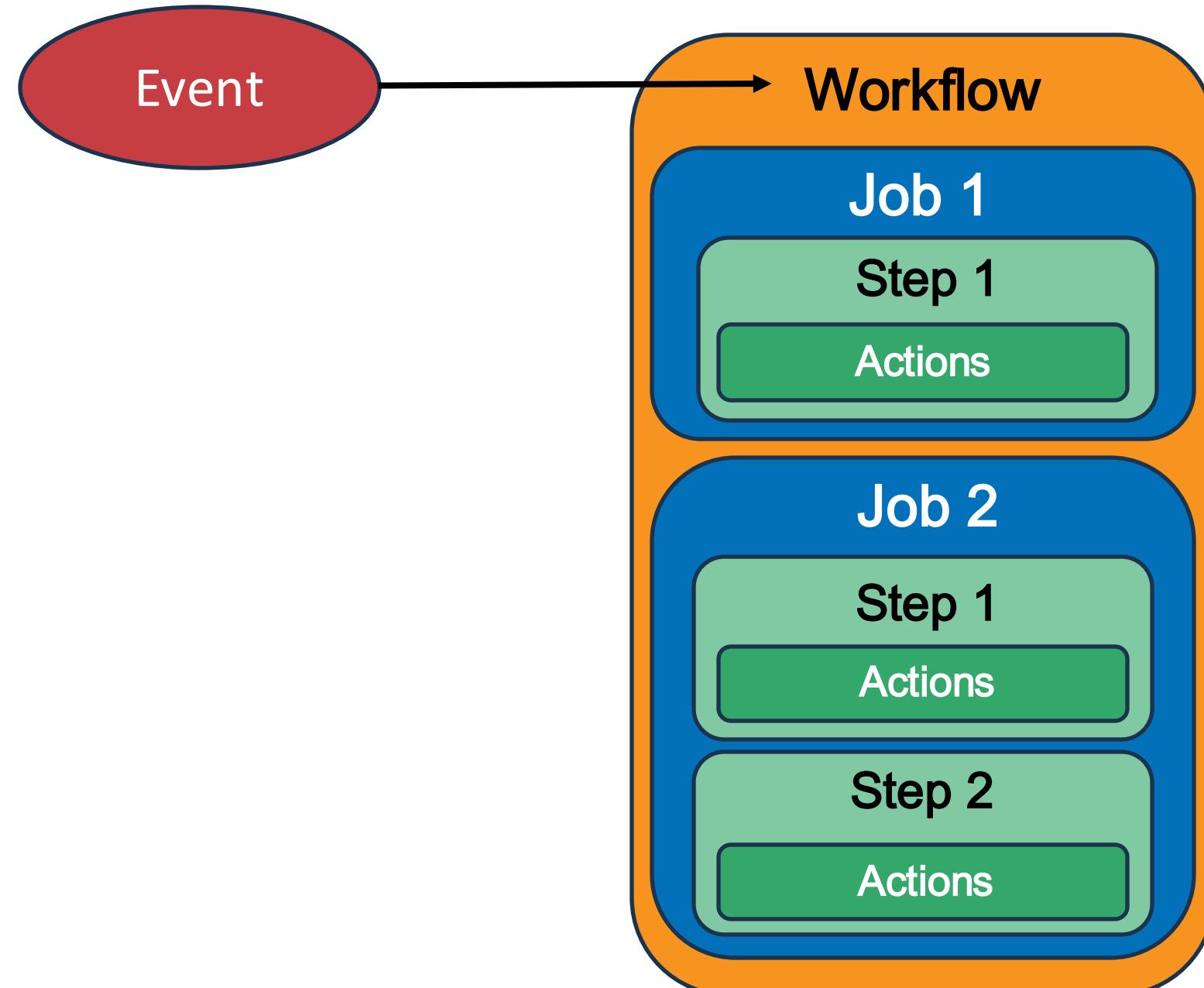
- ✓ Jobs break down further into Steps
- ✓ Steps run sequentially, each performing a single action
- ✓ Examples: checkout code, login to Docker, run tests



Action

- ✓ Actions are reusable building blocks in workflows
- ✓ Choose from GitHub, community marketplace, or create custom
- ✓ Examples: `actions/checkout`, `docker/build-push-action`

Core Concepts of GitHub Actions



Core Concepts of GitHub Actions

```
# -----
# 1. WORKFLOW
# This whole file defines a workflow called "DevOps CICD Pipeline" Stored under .github/workflows/
# -----
name: DevOps CICD Pipeline

# -----
# 2. EVENT (Trigger)
# The workflow runs "on: push" → whenever code is pushed
# -----
on: push

jobs:
  # -----
  # 3. JOB
  # One job called "devops-ci" runs on Ubuntu
  # -----
  devops-ci:
    runs-on: ubuntu-latest

steps:
  # -----
  # 4. STEP
  # Each "name:" entry below is a step, Steps run sequentially inside the job
  # -----
  - name: Checkout
    # -----
    # 5. ACTION
    # This step uses a prebuilt action from GitHub Marketplace
    # -----
    uses: actions/checkout@v5.0.0
  - name: Build and push Docker images
    uses: docker/build-push-action@v6.18.0
    with:
      file: ./Dockerfile
      tags: yogeshraheja/devopspython:${{ github.sha }}
      push: true
```

Demo

DevOps | Understanding CI/CD Automation Tasks

Demo

DevOps | Automating CI Pipelines (CI Workflow)

Demo

DevOps | Installing self-hosted Runners

Demo

DevOps | Automating CD Pipelines (CD Workflow)

Demo

DevOps | Executing end to end CI-CD Pipelines

Demo

DevOps | Updating ArgoCD Progress in GitHub Projects

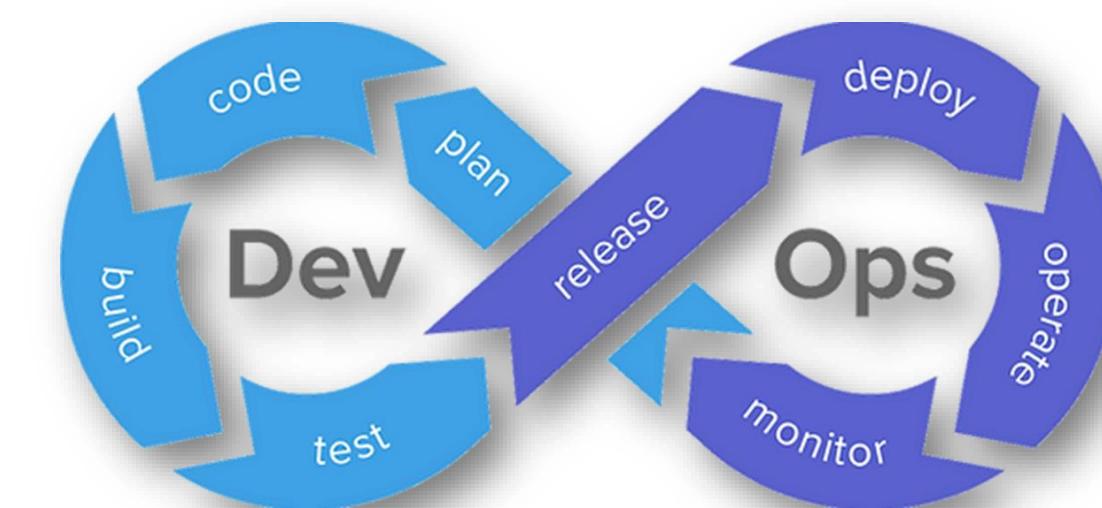


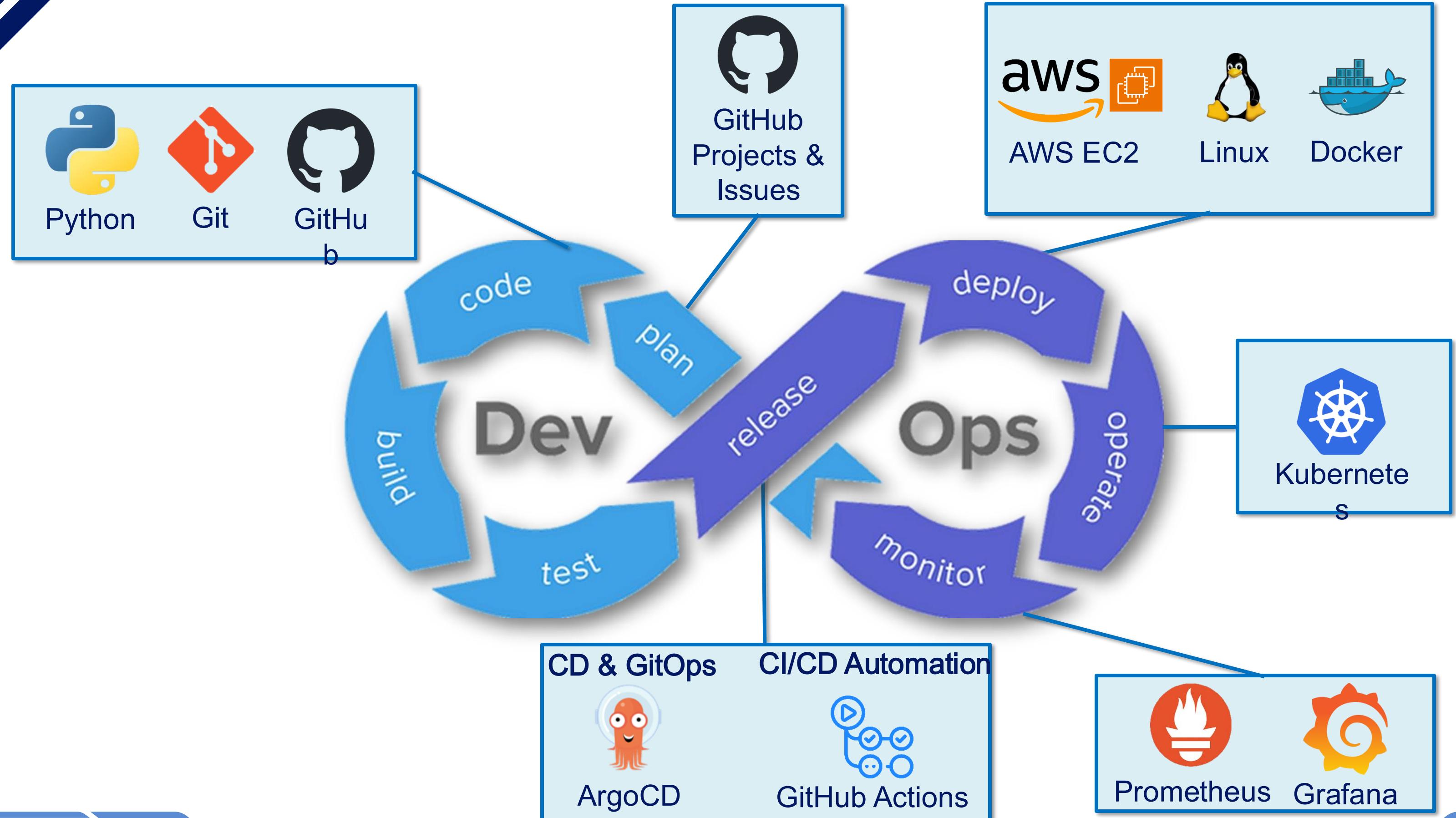
THANK YOU



Practical DevOps

Bootcamp for All







Follow us on:

 @thinknyx

 @thinknyx

 @thinknyx-technologies

 @thinknyx

 @thinknyx-technologies