Containerization Fast Track

WELCOME!

Develop Intelligence

Allen Sanders
Senior Technology Instructor

Join Us in Making Learning Technology Easier

Develop Intelligence

Our mission...

Over 16 years ago, we embarked on a journey to improve the world by making learning technology easy and accessible to everyone.

...impacts everyone daily.

And it's working. Today, we're known for delivering customized tech learning programs that drive innovation and transform organizations.

In fact, when you talk on the phone, watch a movie, connect with friends on social media, drive a car, fly on a plane, shop online, and order a latte with your mobile app, you are experiencing the impact of our solutions.

Copyright DevelopIntelligence LLC

Upskilling and Reskilling Offerings

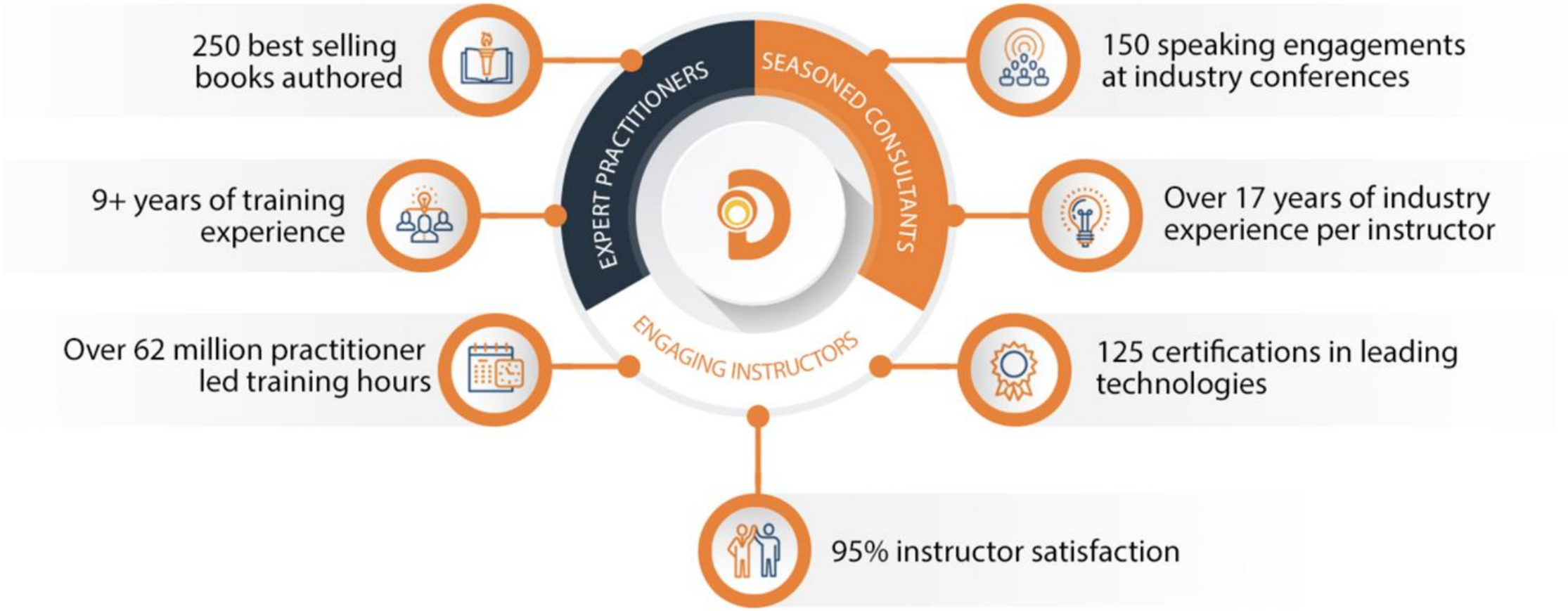
Develop Intelligence

Intimately customized learning experiences just for your teams.

- **Workshop** — 2-3 day upskilling experiences
- **Fast Track** — 5-day reskilling experiences
- **Learning Spike** — 1-day technology overviews
- **Target Topics** — 90-minute instructor-led micro-learnings
- **Hack-a-thon** — Learn and build an MVP in 2-3 days

Copyright DevelopIntelligence LLC

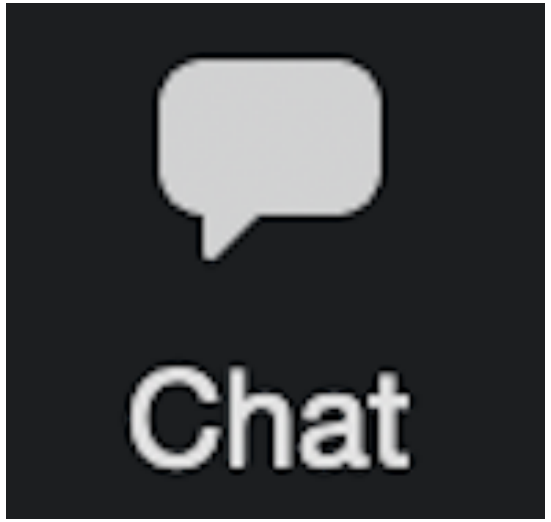
World Class Practitioners



Virtual Training Expectations for You

Arrive on time / return on time

Mute unless speaking



Use chat or ask questions verbally

Virtual Training Expectations for Me

I pledge to:

- Make this as interesting and interactive as possible
- Ask questions in order to stimulate discussion
- Use whatever resources I have at hand to explain the material
- Try my best to manage verbal responses so that everyone who wants to speak can do so
- Use an on-screen timer for breaks so you know when to be back

Prerequisites

- This course assumes you have some experience as an engineer in a modern development shop, and are familiar with the concepts of full stack development, the Software Development Lifecycle, CI/CD and virtual machines

Objectives

At the end of this course, you will be able to:

- Evaluate the challenges of building and running distributed applications and different approaches Monolithic vs Microservice
- Apply best practices for secure, resilient, and self-healing Docker containers that are ready for production
- Explain the Docker core components and architecture and how it integrates with Kubernetes
- Apply Kubernetes system architecture and declarative approach to services, applications, storage, and networking
- Deploy an application on Kubernetes in a local development environment and in the cloud (AWS)

Agenda

- Demystifying and Defining Distributed Systems, Cloud Native, and Infrastructure as Code
- Introduction to Containerization, Docker, and Linux
- Diving Deeper into Docker and Containers
- Introduction to Kubernetes and Container Orchestration
- Diving Deeper into Kubernetes
- Review

Why Containerization?

- Abstracts away the code implementation so you can deploy in a platform-agnostic manner, writing in the language of your choice
- Aligns strongly with the principles and practices of DevOps
- Helps leverage the power of the cloud
- Speeds up important non-coding activities (infrastructure spin-up, testing, CI/CD tasks, DevSecOps, code quality checks, etc.
- Helps breed consistency vs. “snowflake”

Structure of the Course / Course Takeaways

- We will be doing most practical labs on our own machines
- We will use Github and AWS for certain cloud-based labs

Demystifying and Defining Distributed Systems, Cloud Native, and Infrastructure as Code

Develop Intelligence

What Do We Mean by Application Hosting?

- The target infrastructure and runtime platform that will be employed for deployment and execution of an application or system
- Can include compute (CPU and server resources), storage, network, data and operating system

Application Hosting – An “Interesting” Example?

Develop Intelligence

Here’s an example of someone thinking “outside-of-the-box” when it comes to application hosting!

https://mashable.com/article/pregnancy-test-doom/

What Are the Hosting Options with Cloud?

Develop Intelligence

- IaaS
- PaaS
- Serverless / FaaS
- SaaS
- Containers
- What do they all mean?

Infrastructure-as-a-Service (IaaS)

- Involves the building out (and management) of virtual instances of:
  - Compute
  - Network
  - Storage
- Akin to spinning up a server (physical or virtual) in your location or data center complete with disks and required network connectivity
- The difference is in the where – instead of in your data center, it is created in a data center managed by one of the public Cloud providers
- Your organization is responsible for patching the OS, ensuring all appropriate security updates are applied and that the right controls are in place to govern interaction between this set of components and other infrastructure

Platform-as-a-Service (PaaS)

- Involves leveraging managed services from a public Cloud provider
- With this model, an enterprise can focus on management of their application and data vs. focusing on management of the underlying infrastructure
- Patching and security of the infrastructure used to back the managed services falls to the CSP (Cloud Service Provider)
- Many managed services support automatic scale up or down depending on demand to help ensure sufficient capacity is in place
- Part of what is often termed the “Shared Responsibility Model”

Serverless / Functions-as-a-Service (FaaS)

Develop Intelligence

- Also represents a type of managed service provided by the CSP
- Cost structure is usually consumption-based (i.e. you only pay for what you use)
- Supports many different coding paradigms (C#/.NET, NodeJS, Python, etc.)
- Typically, with Serverless (and PaaS), the consumer is only concerned with the application code and data – elements of the CSP's "backbone" used to support are managed by the CSP
- Includes more sophisticated automated scaling capabilities – built for Internet scale

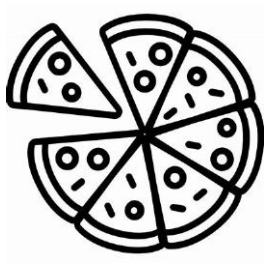
Software-as-a-Service (SaaS)

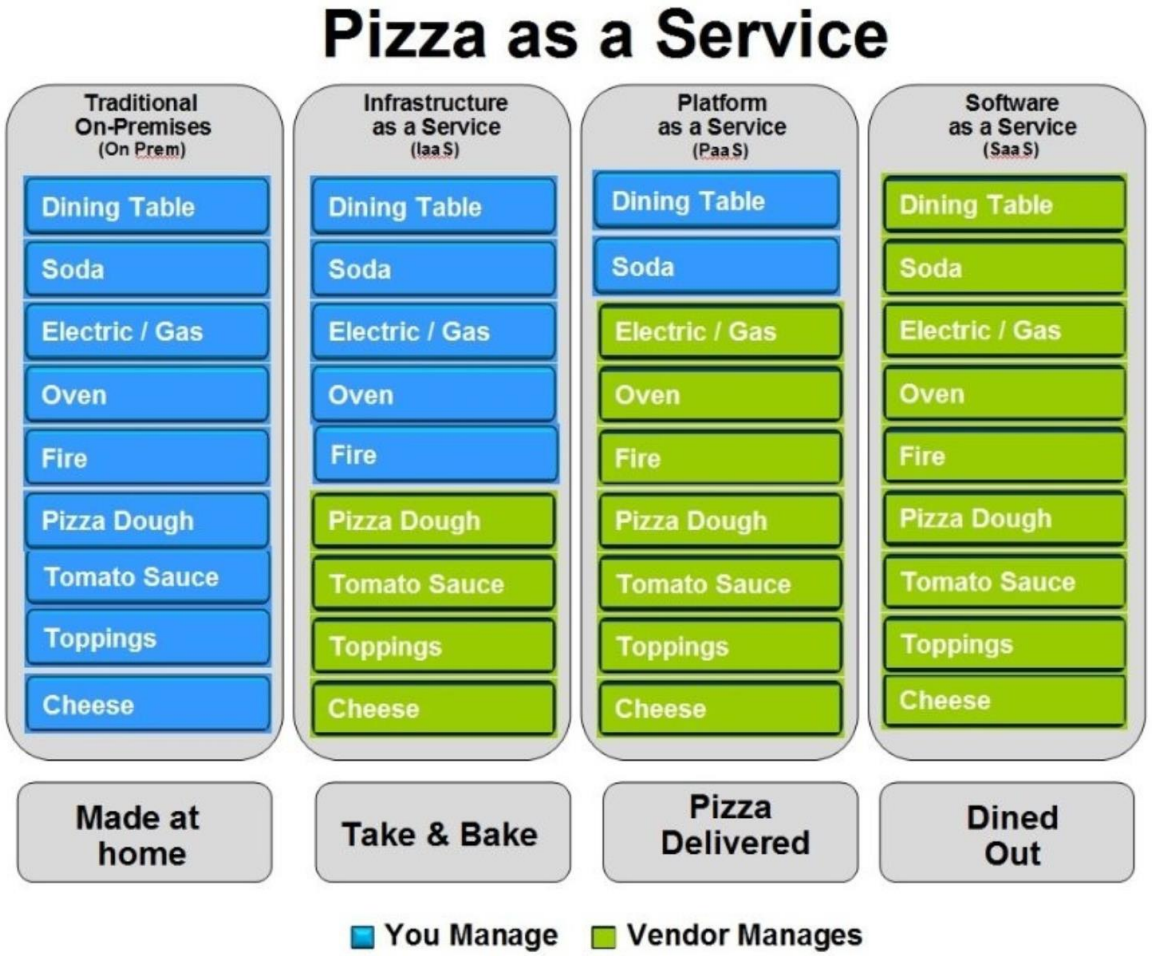
Develop Intelligence

- Subscription-based application services
- Licensed for utilization over the Internet / online rather than for download and install on a server or client machine
- Fully-hosted and fully-managed by a 3rd party
- Of those discussed, often the cheapest option for service consumers
- However, also offers minimal (or no) control, outside of exposed configuration capabilities

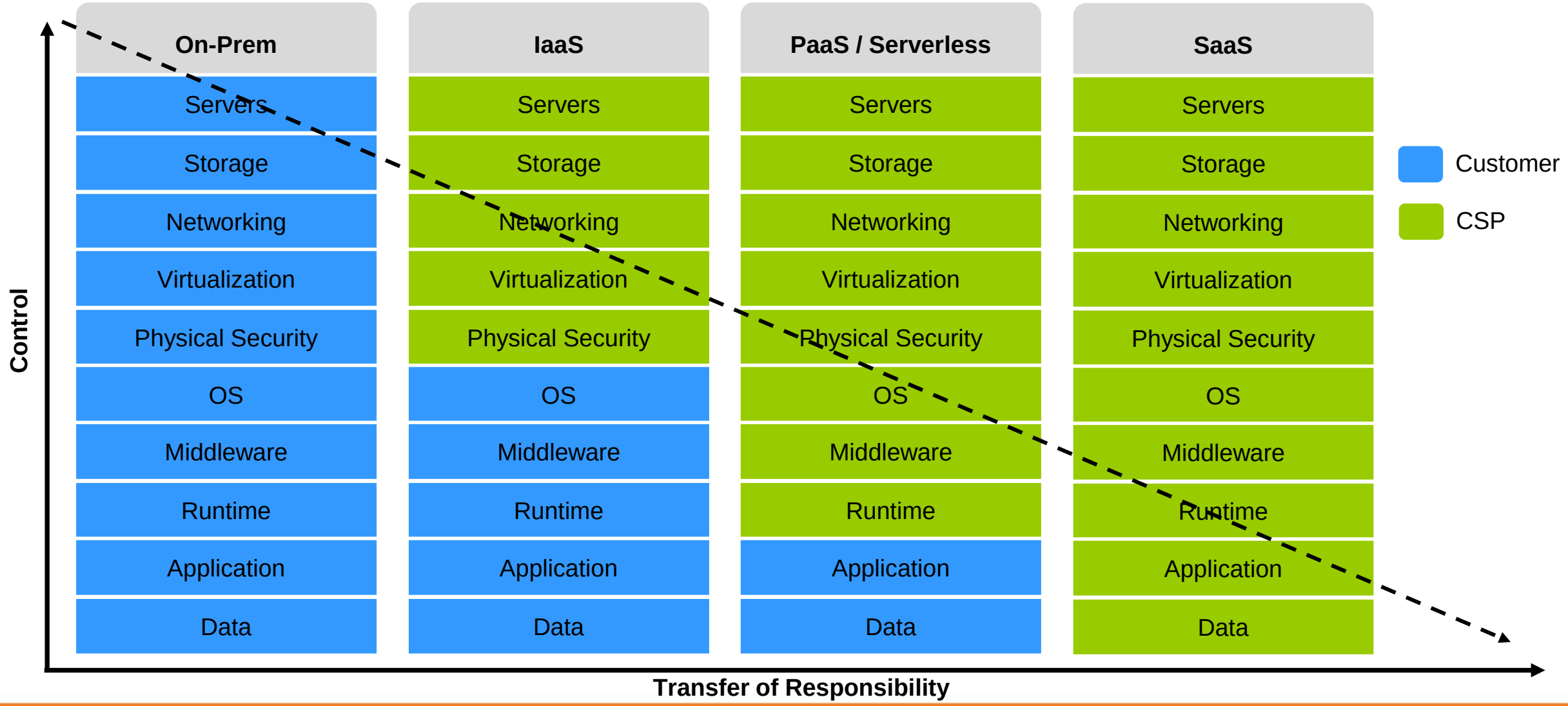
Pizza-as-a-Service

From a LinkedIn post by Albert Barron from IBM (https://www.linkedin.com/pulse/20140730172610-9679881-pizza-as-a-service/)





Side-by-Side Comparison



Containers

- Form of virtualization at the app packaging level (like virtual machines at the server level)
- Isolated from one another at the OS process layer (vs VM's which are isolated at the hardware abstraction layer)
- Images represent the packaging up of an application and its dependencies as a complete, deployable unit of execution (code, runtime and configuration)

Containers

- A platform (e.g., Docker) running on a system can be used to dynamically create containers (executable instances of the app) from the defined image
- Typically, much, much smaller than a VM which makes them lightweight, quickly deployable and quick to “boot up”
- An orchestration engine (e.g., Kubernetes) might be used to coordinate multiple instances of the same container (or a “pod” of containers) to enable the servicing of more concurrent requests (scalability)

What are Microservices?

Develop Intelligence

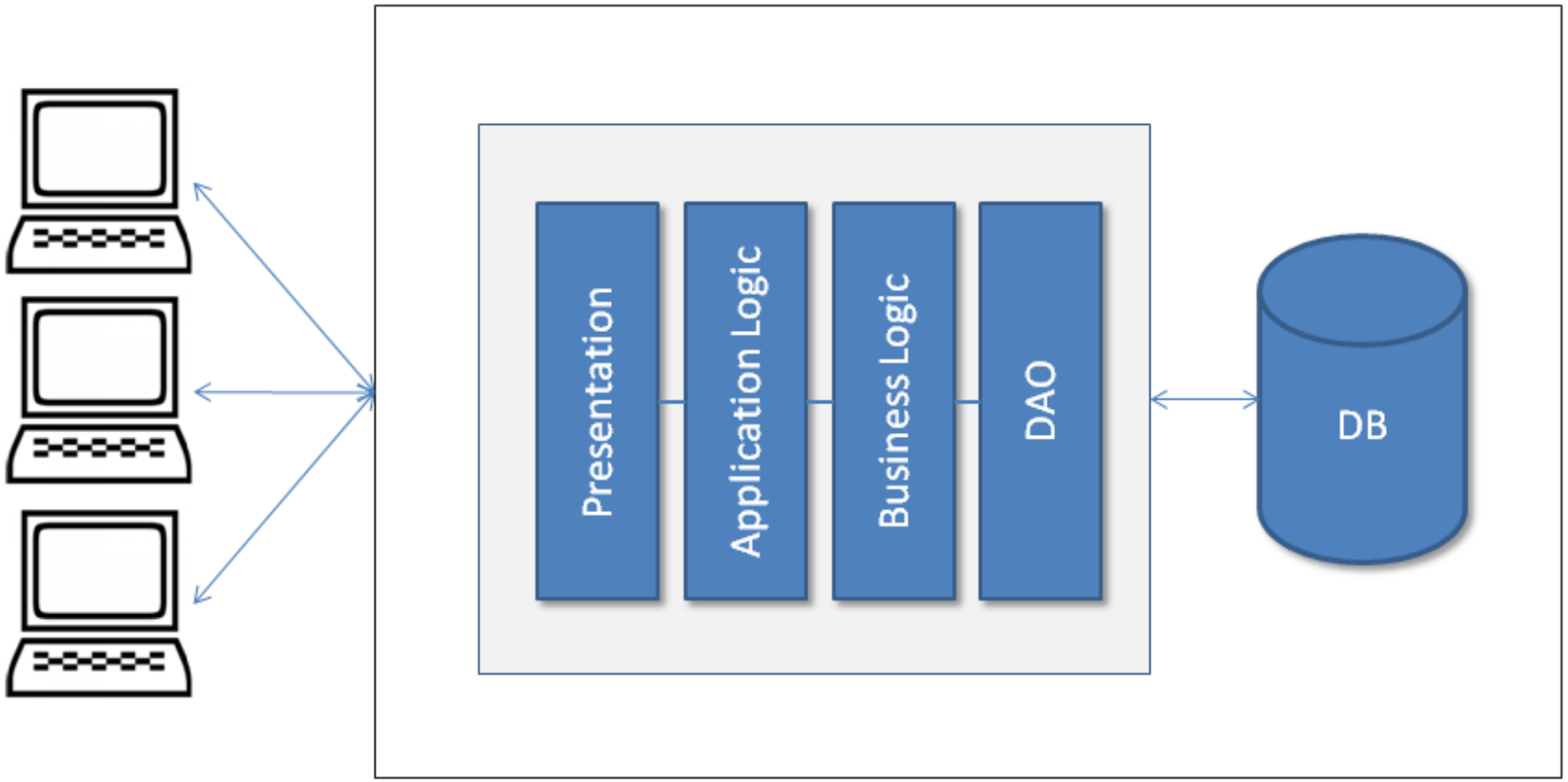
- An architectural style in which a distributed application is created as a collection of services that are:
  - Highly maintainable and testable
  - Loosely coupled
  - Independently deployable (by extension, independently-scalable)
  - Domain centric and organized around business capabilities
- The Microservices architecture enables the continuous deployment and delivery of large, complex distributed applications

What is a Monolith?

Develop Intelligence

- Typical enterprise application
- Large codebases
- Set technology stack
- Highly coupled elements
- Whole system affected by failures
- Scaling requires the duplication of the entire app
- Minor changes often require full rebuild

Monolithic Architecture Example

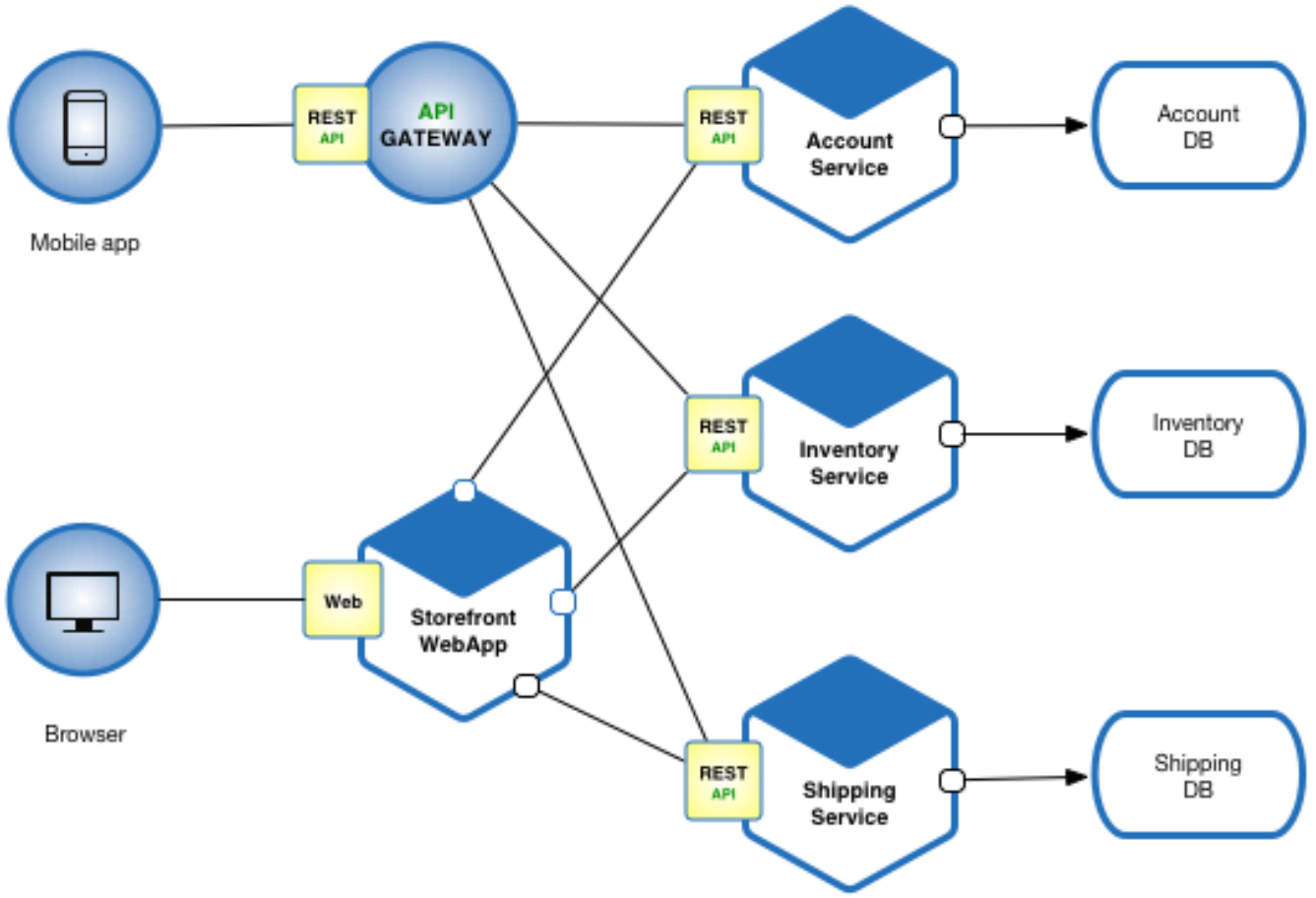


Moving to Microservices

Develop Intelligence

- We have moved towards a refinement of “what is a service?”
- Even in SOA, we can create a service that does more than one thing – using it to solve an overall business problem, rather than one part of the actions needed to solve that problem
  - This was moving us back towards monolithic
- We arrive at a point where we want individual, purposeful services that do one thing and do it well – the microservice

Example Microservices Architecture



Microservices – Benefits vs. Costs

Develop Intelligence

Benefits:

- Enables work in parallel
- Promotes organization according to business domain
- Advantages from isolation
- Flexible in terms of deployment and scale
- Flexible in terms of technology

Microservices – Benefits vs. Costs

Develop Intelligence

Costs:

- Requires a different way of thinking
- Complexity moves to the integration layer
- Organization needs to be able to support re-org according to business domain (instead of technology domain)
- With an increased reliance on the network, you may encounter latency and failures at the network layer
- Transactions must be handled differently (across service boundaries)

Microservices & Good Architecture

Develop Intelligence

- With microservices, key concepts:
  - Cohesion (goal to increase)
  - Coupling (goal to decrease)

- SOLID principles & DDD (Domain Driven Design) lay the foundation for:
  - Clean, logical organization of components
  - Maintainability
  - Clear boundaries, encapsulation and separation of concerns in the components used to build out complex systems
  - Techniques that minimize coupling
  - Being “surgical” with our change

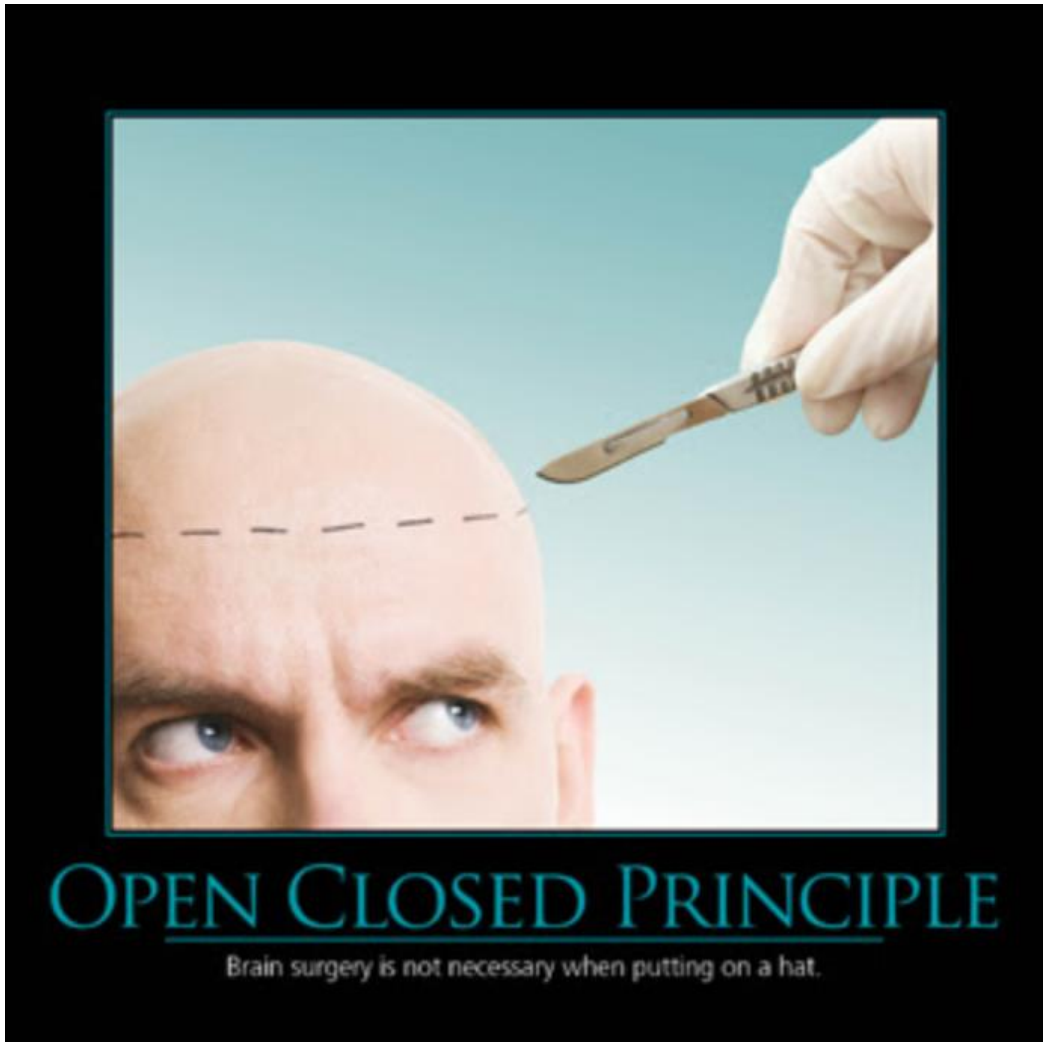
SOLID Principles

SOLID principles help us build testable code

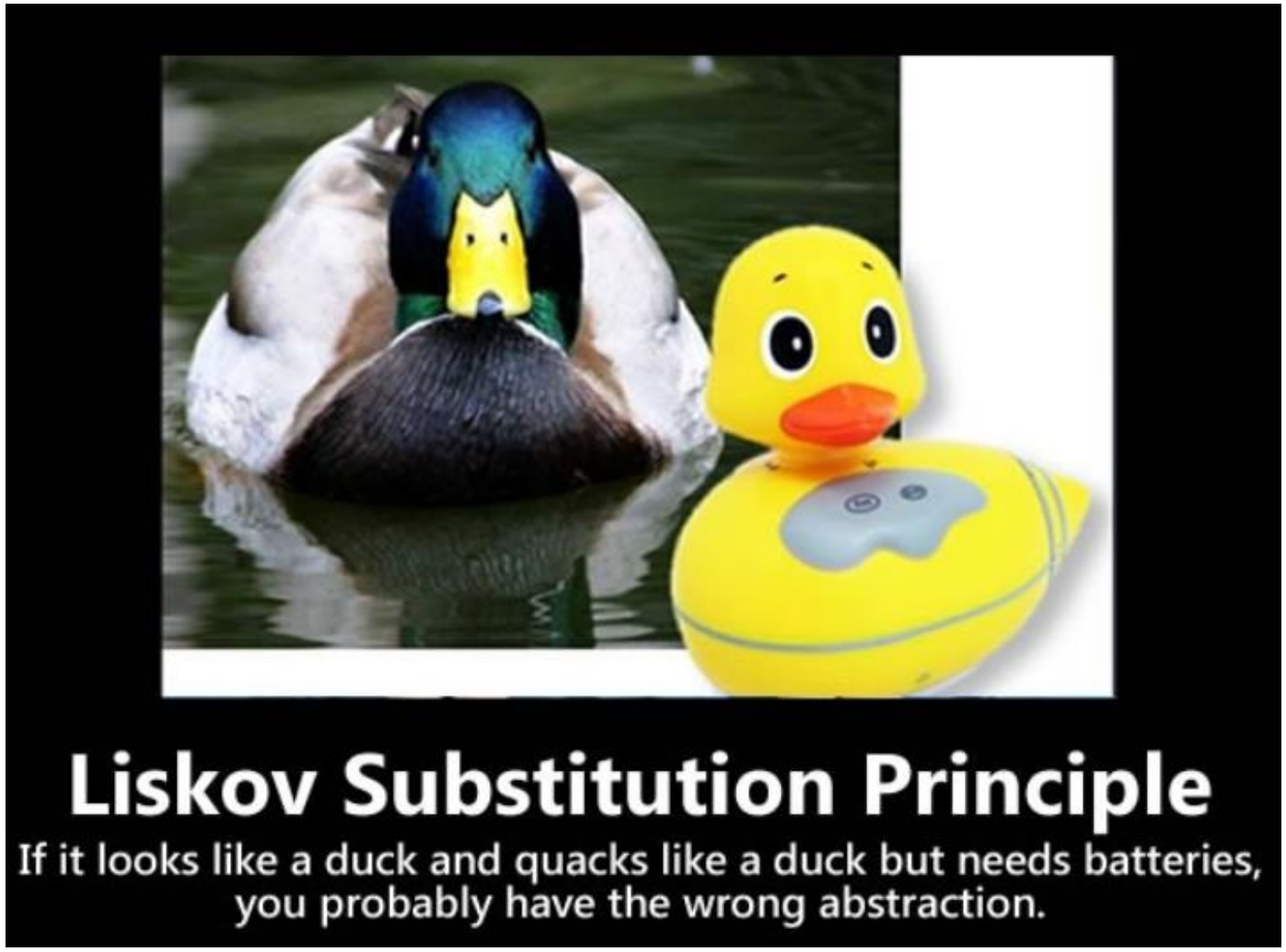
- Single Responsibility Principle (SRP)
- Open-Closed Principle (OCP)
- Liskov Substitution Principle (LSP)
- Interface Segregation Principle (ISP)
- Dependency Inversion Principle (DIP)

Single Responsibility Principle (SRP)

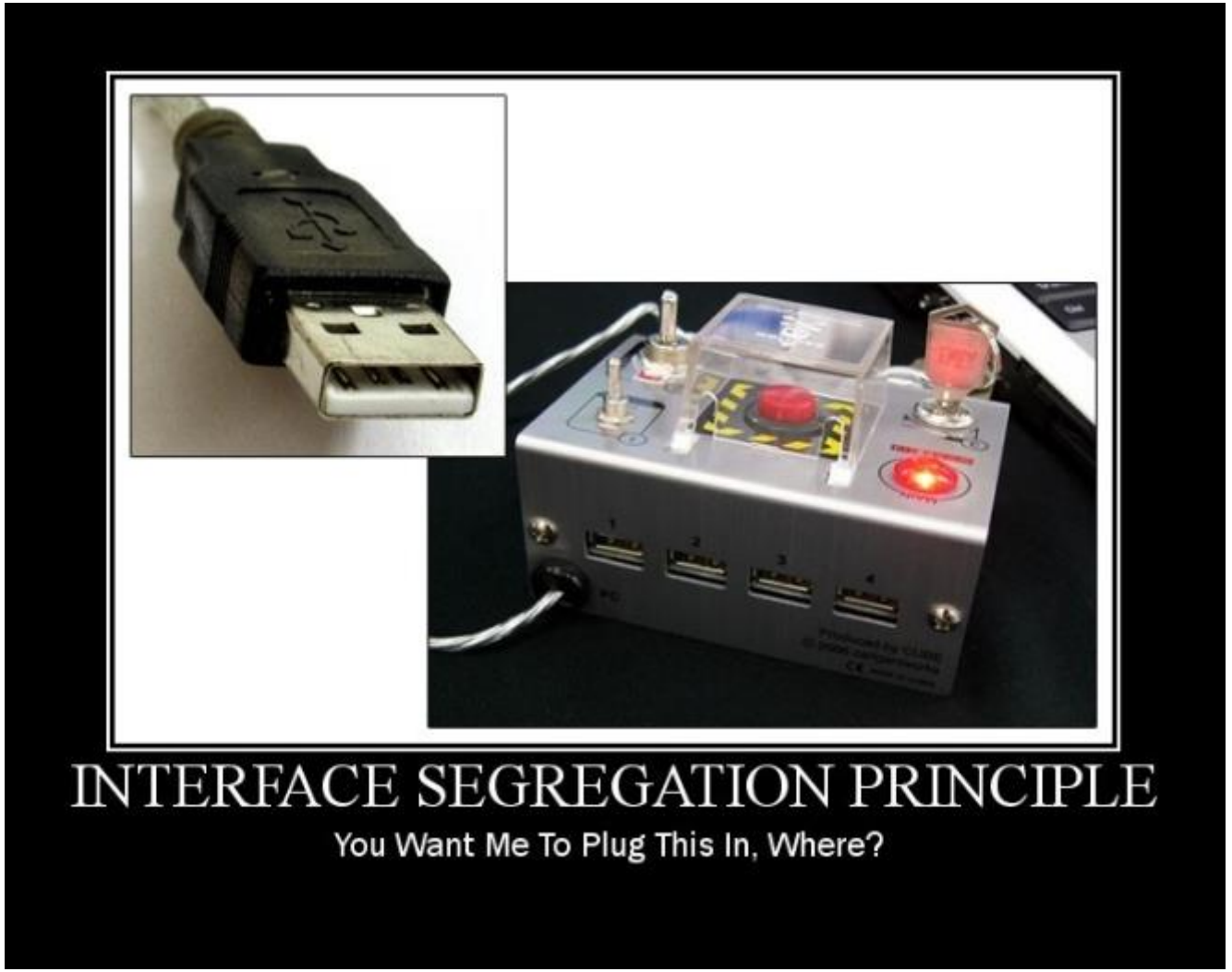
Open-Closed Principle (OCP)



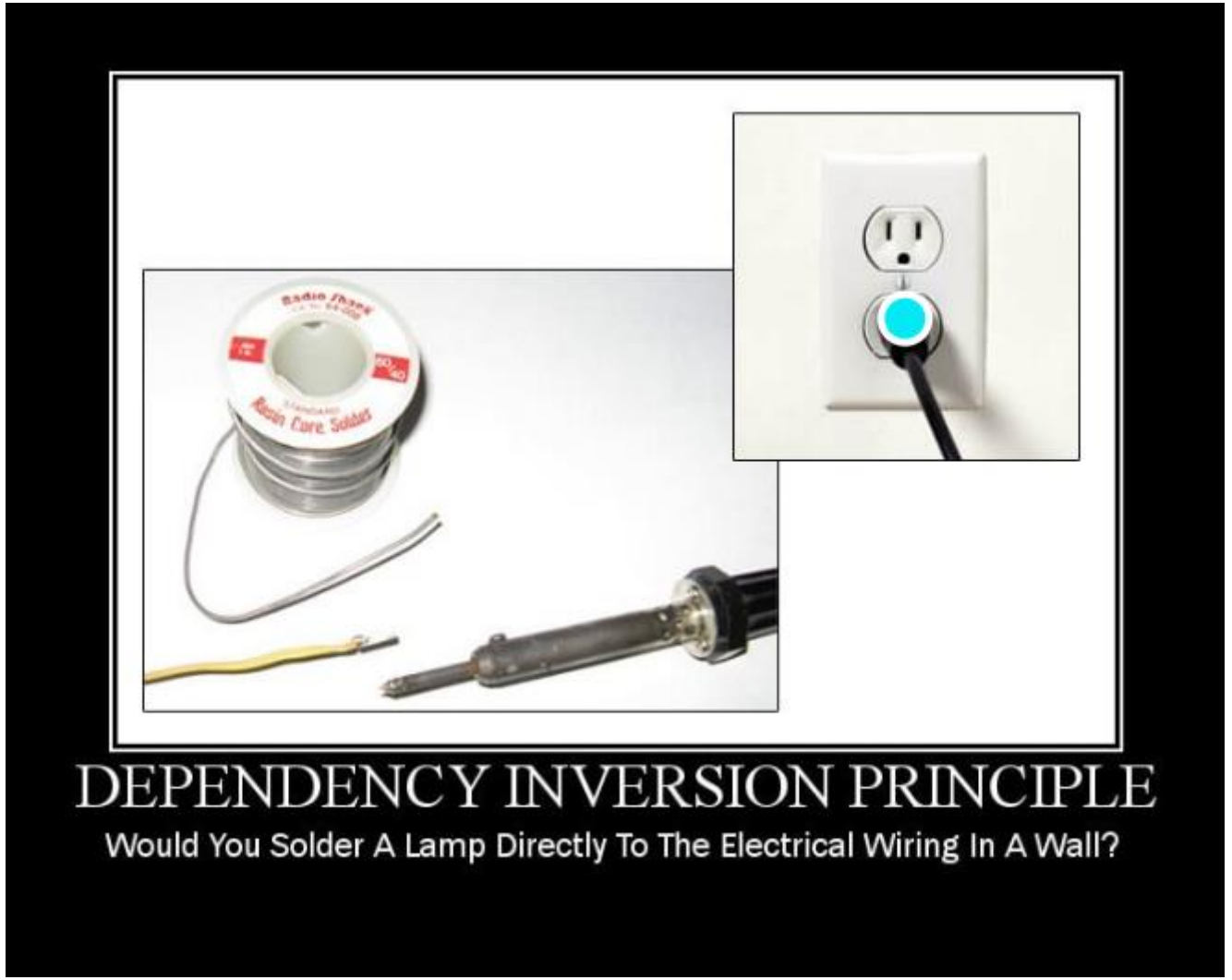
Liskov Substitution Principle (LSP)



Interface Segregation Principle (ISP)



Dependency Inversion Principle (DIP)



Microservices & Containers

Develop Intelligence

- Microservices – with their smaller size, independently-deployable and independently-scalable profile, and encapsulated business domain boundary – are a great fit for containers
- Using Kubernetes, sophisticated systems of integrated microservices can be built, tested and deployed
- Leveraging the scheduling and scalability benefits of Kubernetes can help an organization target scaling across a complex workflow in very granular ways
- This helps with cost management as you can toggle individual parts of the system for optimized performance

Microservices & the Cloud

Develop Intelligence

- Microservices have broad support across multiple Cloud providers
- One option includes standing up VM's (IaaS) and installing / managing a Kubernetes cluster on those machines or
- Another option includes leveraging a managed service (PaaS) provided by the CSP
- Microservices are a great option for the Cloud because the elastic scalability provided by the Cloud infrastructure can directly support the independent scalability needed with a microservices architecture

Microservices Summary

Develop Intelligence

- Applications that can
  - Efficiently scale
  - Are flexible
  - Are high performance
- These apps are powered by multiple services, working in concert
- Each service is small and has a single focus
- We use a lightweight communication mechanism to coordinate the overall system
- This ends up giving us a technology agnostic API

Demo: Microservices Application

Develop Intelligence

Reference Application: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/multi-container-microservice-net-applications/microservice-application-design

Repo: https://github.com/dotnet-architecture/eShopOnContainers

Discussion: Monolith vs. Microservices

- Challenges with monoliths?
- Experience with SOA?
- Experience with Microservices?

Stateless Applications and Scaling

Develop Intelligence

- If state is held in your application, scaling is very hard. Why?
- Where can we get state info if it's not held in the application?
- If we build to resist failure, scaling is hard. Why?
- How can we build to embrace failure?

Modern Approaches to Application Development

- Leverage the cloud
- Abstract away as much of the non-code elements as possible
- Use the tools that arise from DevOps (automation, feedback, etc.)
- Integrate testing and code quality checks early (“Shift Left”)

Design Approaches to Modern Applications

Develop Intelligence

- Decompose when possible
- Embrace Failure
- Separate the language from the function of the code

Cloud-Native

Develop Intelligence

- (From Pivotal): “Cloud-native is an approach to building and running applications that exploits the advantages of the cloud computing delivery model. Cloud-native is about how applications are created and deployed, not where.”
- Can be achieved in both public and private cloud systems.
- Key concepts:
  - DevOps
  - Continuous Delivery
  - Microservices
  - Containers
- Key element is making nearly limitless computing power available on demand.

Infrastructure as Code (IaC)

Develop Intelligence

- Declarative vs. Imperative coding
- Drawbacks of declarative infrastructure
- Impact of IaC on the dev process, on testing, and on deployment

Code Reviews in a Thriving DevOps System

Develop Intelligence

- How often do you face “but it works on my machine”?
- If infrastructure is abstracted, you can see the code in action easily
- You can put your attention on application logic, not on dependencies, runtimes, hardware, configuration, etc.

Introduction to Containerization, Docker, and Linux

Develop Intelligence

What is a Container?

Develop Intelligence

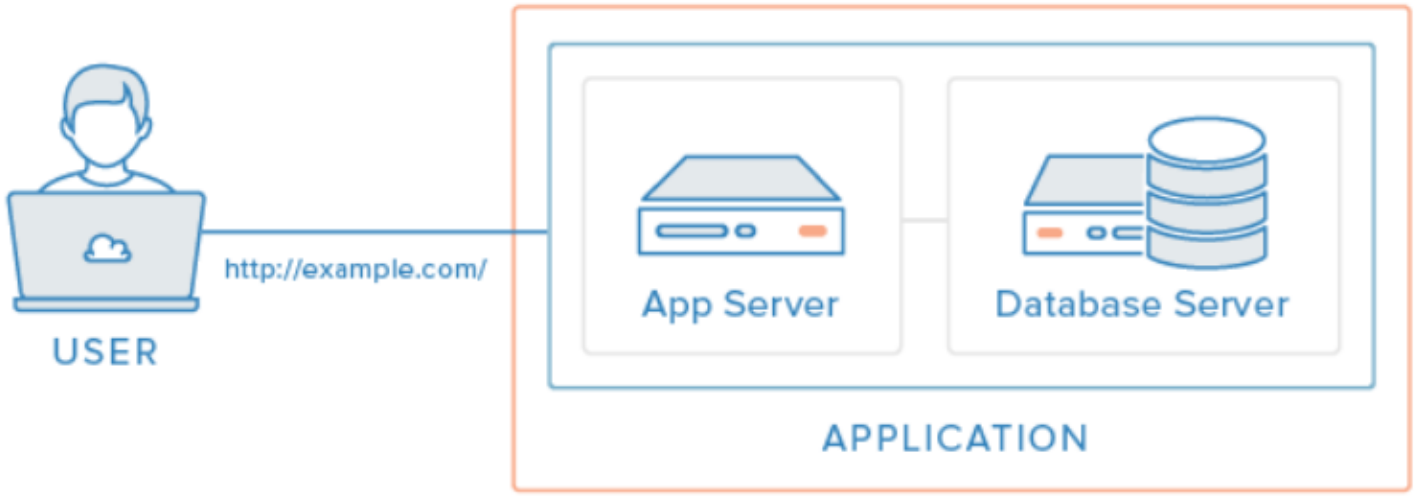
- Loosely isolated environment used to package and deploy an application in a platform-agnostic manner

Evolution of Containers

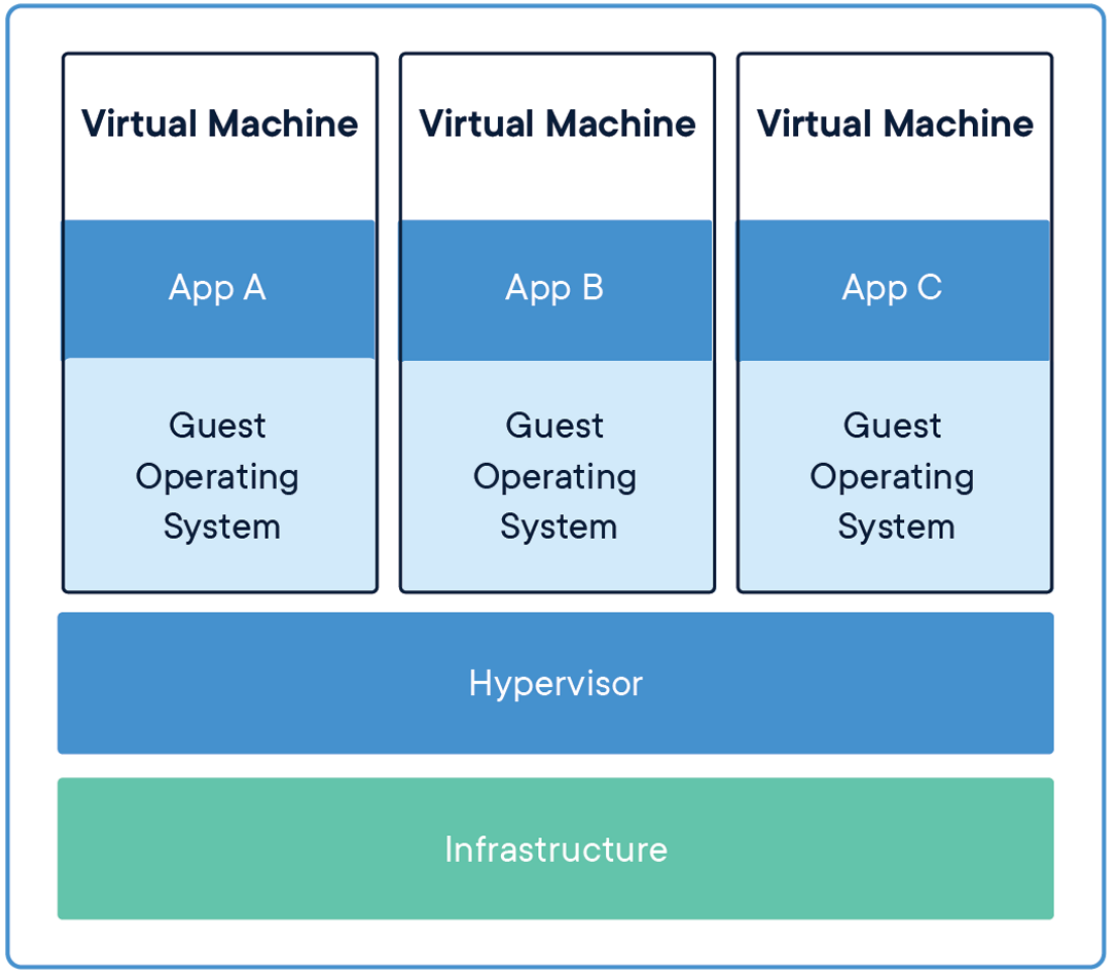
Develop Intelligence

- Started with client/server architecture – dedicated server for each app
- Moved to Virtual Machines – better, but had problems of OS overhead, etc.
- Containers mean we don’t have to worry about specific hardware, OS, language, etc. – we just care about the code

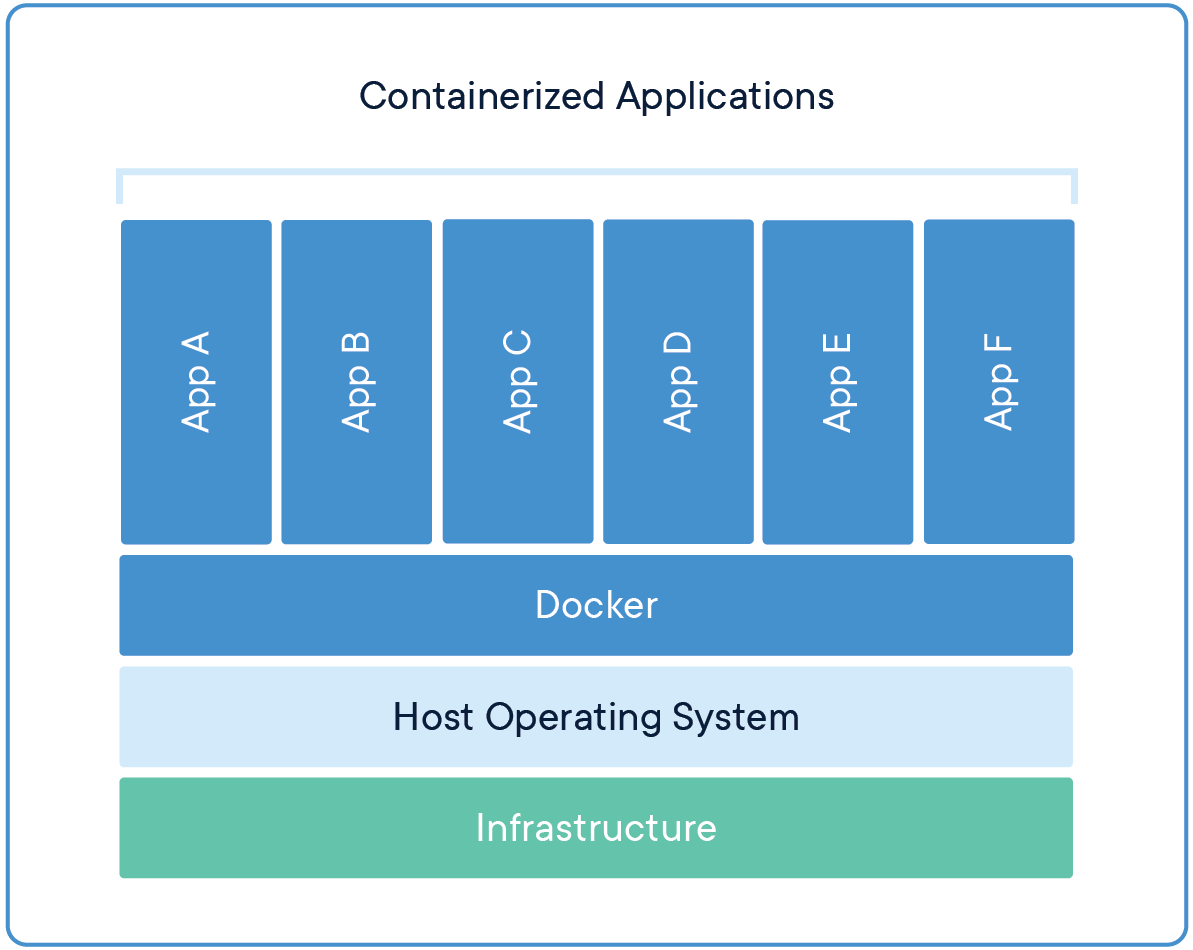
Client/Server



Virtual Machines



Containers


Containerized Applications
App A
App B
App C
App D
App E
App F
Docker
Host Operating System
Infrastructure

The Container Ecosystem

Develop Intelligence

- Docker: The company vs. the application
- Docker client/server architecture
  - Docker daemon (dockerd)
  - Docker Client (docker)
  - Host
  - Registries (public/private)
- Docker Objects
  - Images
  - Containers
  - Networks
  - Volumes
  - Plugins

What is Docker?

Develop Intelligence

- Open-source containerization technology
- Enables deployment of self-contained & isolated application instances
- One of the foundational technologies for supporting microservices

What is Docker?

Develop Intelligence

- Built around the concept of images & containers
- Also, supports composition of a set of containers to be deployed together
- For example, application code + network components + database components

What is Docker?

Develop Intelligence

- Utilizes principles of “immutable infrastructure”
- Complete application environments torn down and recreated as needed
- Helps to minimize infrastructure “drift” and environment inconsistencies

Demo: Creating an Image and a Running Container

Develop Intelligence

- Make a stock ASP.NET Core app
- Demo it in action
- Create the dockerfile
- Create the image
- Start the container
- Demo it in action

Setting Up a Docker Dev Environment

Develop Intelligence

- Tools:
  - Docker Desktop
  - Code editor/IDE that plays nice with containers (VS Code for us)

Lab: Setting up a Local Docker Dev Environment

- Install Docker Desktop
  - Share drives if needed
  - Choose Linux vs. Windows containers (Linux)
  - Choose whether to turn on Kubernetes (not yet)

The Dockerfile

Develop Intelligence

- Tells Docker what to do in creating an image for your application
- The commands are all things you could do from the CLI
- Used by the docker “build” command
- Docker build uses this file and a “context” – a set of files at a specified location – to make your image

Dockerfile example

- The following creates an image for building/running Java app in container
- See https://github.com/KernelGamut32/dockerlab-repo-sample for sample

Dockerfile

```
# Grabs OpenJDK image upon which the new image will be based
FROM openjdk:17

# Creates a new target folder in image
RUN mkdir /usr/src/JavaDemoApp

# Copies current directory contents to newly created folder
COPY . /usr/src/JavaDemoApp

# Switches working directory in image to app folder
WORKDIR /usr/src/JavaDemoApp

# Compiles/builds Java app
RUN javac JavaDemo.java

# Executes new Java app
CMD ["java", "JavaDemo"]
```

Docker Images

- Represent templates defining an application environment
- New instances of the application can be created from the image
- These instances are called containers

Docker Images

- Images are defined via a Dockerfile definition
- Support layers for building up the environment in stages
- Fully defines the application, including all components required to support

Docker Images

- Those components can include:
  - Runtime
  - Development framework
  - Source code
  - Executable instructions for container startup

Docker Images

Develop Intelligence

- Start with a base that gives your app a place to live
  - Needed OS/runtimes/dB server applications, etc.
- Examples:
  - nginx
  - Node
  - MySQL
  - Apache HTTP Server
  - IIS with .NET Runtimes

Docker Hub

Develop Intelligence

- Centralized registry for image storage & sharing
- Can signup for an account – user accounts offer both free and pro versions
- Also, supports organizations for grouping of multiple team members

Docker Hub

- Accessible at https://hub.docker.com
- Search feature enables search for image by technology or keyword
- Image detail displays available tags and image variants

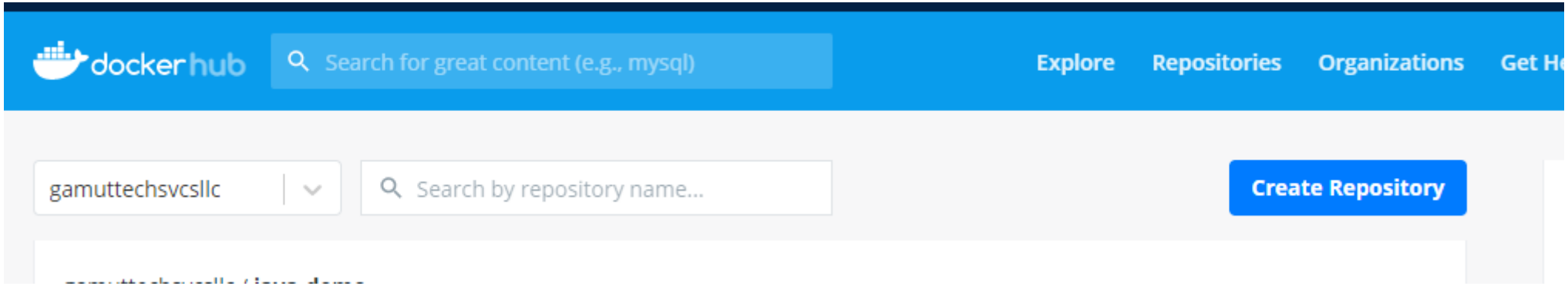
Docker Hub

- May also include examples of usage
- Pro account supports image scanning for security vulnerabilities
- Can be useful for image reuse and image sharing across a dev team

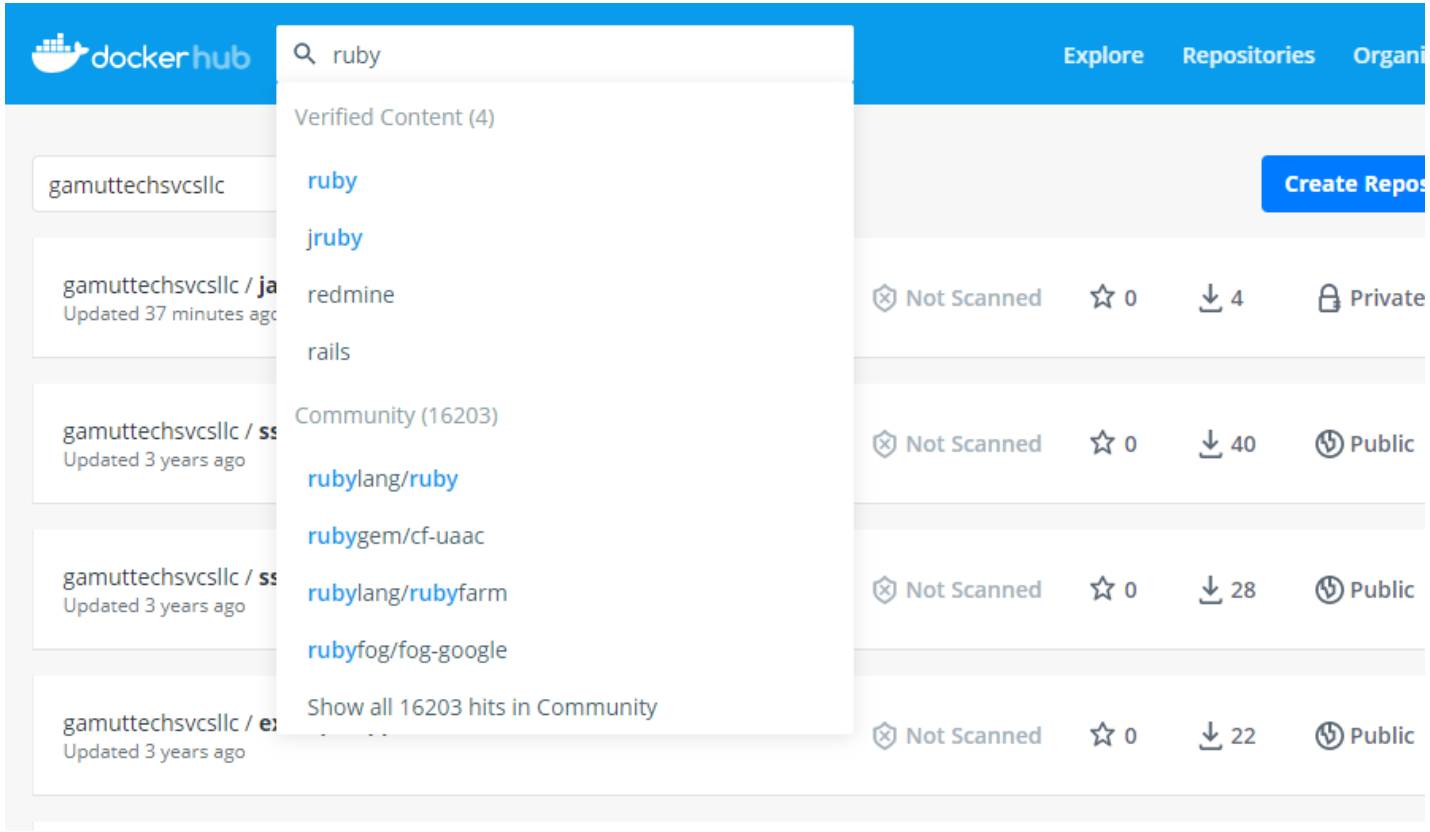
Docker Hub

- There are other registry types, including private registries

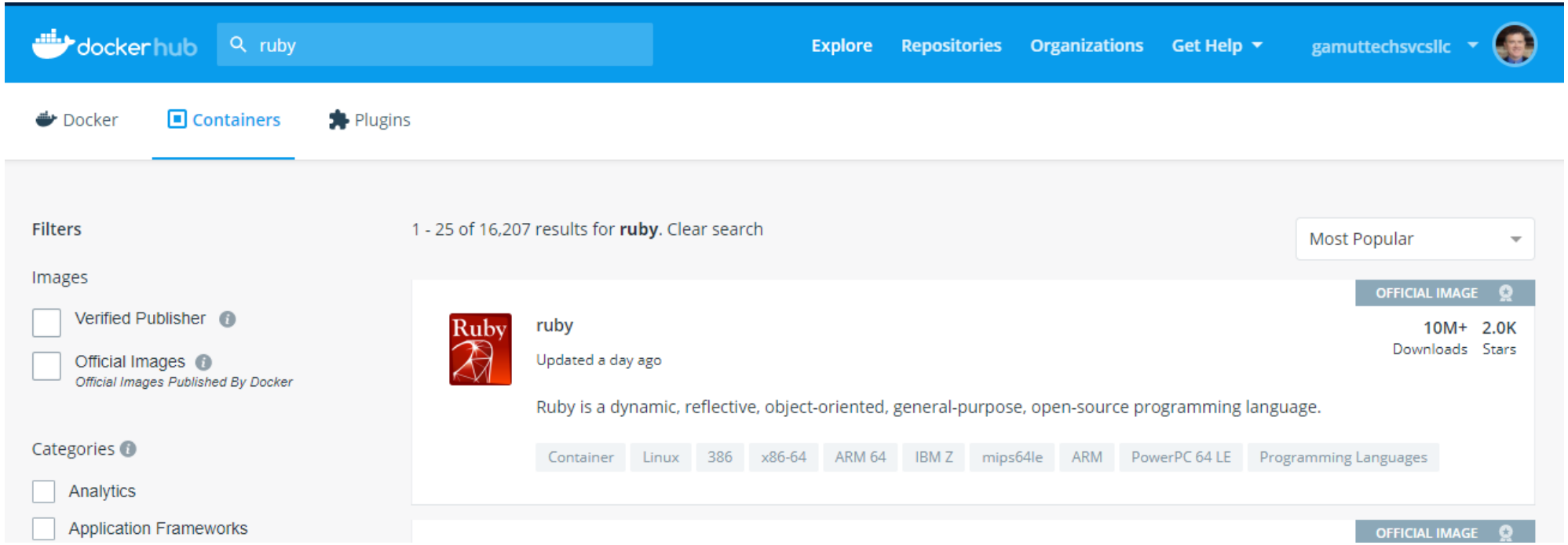
Docker Hub


dockerhub
Search for great content (e.g., mysql)
Explore
Repositories
Organizations
Get H
gamuttechsvcsllc
Search by repository name...
Create Repository

Docker Hub


docker hub
ruby
Explore
Repositories
Organi
Verified Content (4)
ruby
jruby
redmine
rails
Community (16203)
rubylang/ruby
rubygem/cf-uaac
rubylang/rubyfarm
rubyfog/fog-google
Show all 16203 hits in Community
gamuttechsvcsllc
Create Repos
gamuttechsvcsllc / ja
Updated 37 minutes ago
Not Scanned
0
4
Private
gamuttechsvcsllc / ss
Updated 3 years ago
Not Scanned
0
40
Public
gamuttechsvcsllc / ss
Updated 3 years ago
Not Scanned
0
28
Public
gamuttechsvcsllc / ex
Updated 3 years ago
Not Scanned
0
22
Public

Docker Hub


docker hub
ruby
Explore
Repositories
Organizations
Get Help
gamuttechsvcsllc
Docker
Containers
Plugins
Filters
1 - 25 of 16,207 results for ruby. Clear search
Most Popular
Images
Verified Publisher
Official Images
Official Images Published By Docker
OFFICIAL IMAGE
ruby
Updated a day ago
10M+ Downloads
2.0K Stars
Ruby is a dynamic, reflective, object-oriented, general-purpose, open-source programming language.
Container
Linux
386
x86-64
ARM 64
IBM Z
mips64le
ARM
PowerPC 64 LE
Programming Languages
Categories
Analytics
Application Frameworks
OFFICIAL IMAGE

Docker Hub

ruby ☆

Docker Official Images

Ruby is a dynamic, reflective, object-oriented, general-purpose, open-source programming language.

100M+

Container | Linux | PowerPC 64 LE | 386 | x86-64 | ARM 64 | IBM Z | mips64le | ARM | Programming Languages

Official Image

Copy and paste to pull this image

```
docker pull ruby
```

View Available Tags

Description | Reviews | Tags

Docker Hub

Develop Intelligence

How to use this image

Create a `Dockerfile` in your Ruby app project

```
FROM ruby:2.5

# throw errors if Gemfile has been modified since Gemfile.lock
RUN bundle config --global frozen 1

WORKDIR /usr/src/app

COPY Gemfile Gemfile.lock ./
RUN bundle install

COPY . .

CMD ["./your-daemon-or-script.rb"]
```

Put this file in the root of your app, next to the `Gemfile`.

You can then build and run the Ruby image:

```
$ docker build -t my-ruby-app .
$ docker run -it --name my-running-script my-ruby-app
```

Generate a `Gemfile.lock`

Demo: Tour through the Docker Registry

Develop Intelligence

https://hub.docker.com/search?q=&type=image&category=base

Building The Image

Develop Intelligence

- To build the image from Dockerfile use *docker build*
- *docker build -t <tag name> <path to Dockerfile>*
- For example, *docker build -t java-demo .*
- Builds image from Dockerfile in current folder (.) with tag name “java-demo”

Building The Image

- For tag name, can include optional detail:
  - Docker ID in Docker Hub for eventual push to image registry
  - Version identifier for tag – defaults to “latest” if excluded
- For example, *docker build -t <docker ID>/<tag name>:<version> .*

Pushing/Pulling Image

Develop Intelligence

- Use *docker push <docker ID>/<tag name>:<version>* to push to private registry
- Use *docker pull <docker ID>/<tag name>:<version>* to pull from private registry
- May prompt for credentials (e.g., Docker Hub login)

Pushing/Pulling Image

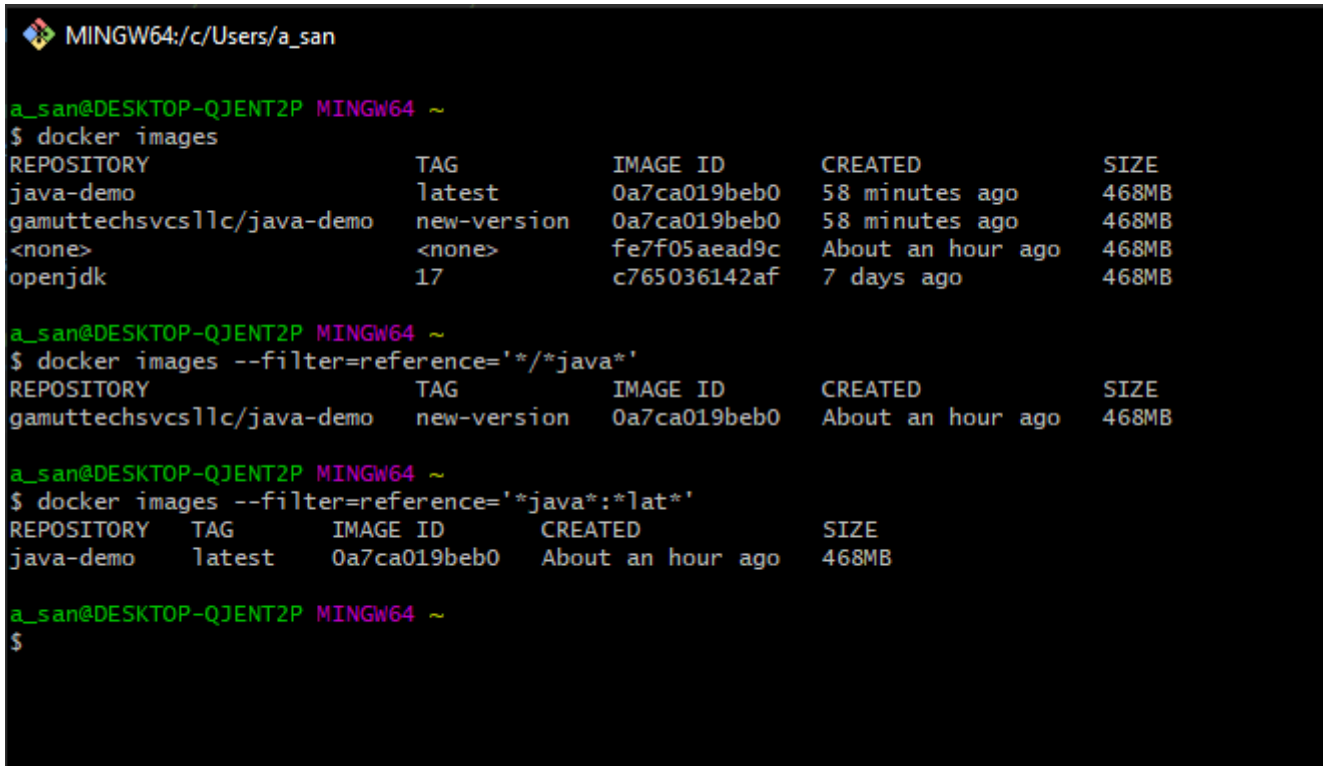
Develop Intelligence

- To pull from public registry, use docker pull <tag name>:<version>
- For example, *docker pull openjdk:17*
- A .dockerignore file can be used to omit files/folders on push

Managing Images

- Use *docker images* to list available local images
- --*filter* argument enables wildcard search
- *docker images --filter=reference='<wildcard for tag name>:<wildcard for version>'*

Managing Images



Managing Images

Develop Intelligence

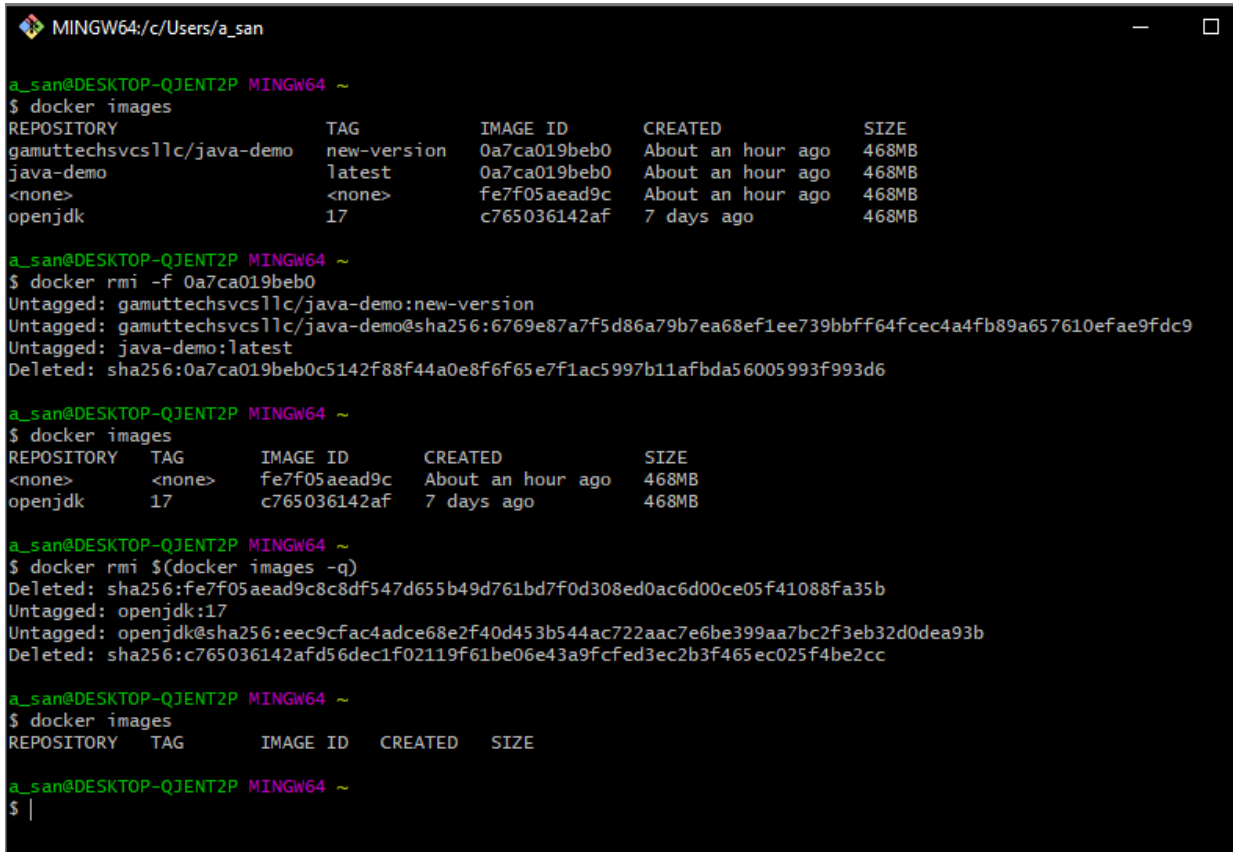
- To remove an image, use *docker rmi*
- *-f* argument used to remove images even when used by containers
- Can use *docker rmi -f <image ID>* to remove specific image

Managing Images

Develop Intelligence

- Can use *docker rmi -f $(docker images -q)* to remove all (CAUTION)
- *docker images -q* lists images in quiet mode (returns image ID's only)

Managing Images



Layers in a Docker Image

Develop Intelligence

- Each instruction in a dockerfile makes a “layer”
- It’s actually a diff from the earlier layer
- Allows Docker to skip redundant info and use cached artifacts
- In this way, images themselves are actually diffs – they show what changed from the earlier stage (e.g., our app’s image is actually a diff from the base image you chose)

Lab: Creating a Docker Image

Develop Intelligence

Best Practices for Creating Docker Images

Develop Intelligence

- Single app per container
- Don’t include unnecessary tools in your image (dev tools; network tools like netcat, etc.)
- Build as small an image as possible
  - Choose a small base image
  - Optimize your app for image size
- Use a consistent “tag” strategy
  - Document it
  - Use it for version info, testing strategy info, etc.
- Be smart about using public base images

Troubleshooting Docker Images

Develop Intelligence

- Most common area is the Dockerfile
- Error on docker build has good messaging and an error code

Docker Containers

Develop Intelligence

- Represent “runnable” instances of a docker image
- Application instance created from image in container can be used as:
  - Isolated executable
  - Request servicer (e.g., web listener)

Docker Containers

Develop Intelligence

- Includes all dependencies and runtime defined by the image
- Isolated from other containers at the OS process layer
- Mechanisms exist to share resources across containers (e.g., files or DBs)
- However, isolation is what makes them powerful – avoid unnecessary coupling

Docker Containers

Develop Intelligence

- Typically, containers are much smaller which makes them lightweight
- Quickly deployable and quick to “boot up”
- Isolation allows technologies like k8s to spin up multiple as needed
- “Load balancers” route to any of multiple instances using single point of connection

Creating Containers

- To create a new container, you can use *docker create*
- Multiple options are provided for configuring container (see *docker create --help*)
- For example, *docker create --name <container name> <image tag>*
- *<image tag>* defines image (template) from which to create container

Creating Containers

Develop Intelligence

- To list available containers, use *docker ps*
- *docker ps -a* lists all containers (even those not currently started)
- Containers can be stopped and started

Creating Containers

Develop Intelligence

- Use *docker start <name>* or *docker start <container ID>* to start
- Use *docker stop <name>* or *docker stop <container ID>* to stop
- Use *docker run* to create and start container in single step
- *docker run* is the more common command

Configuring Containers

Develop Intelligence

- Command-line options for configuring containers using *docker run* include:
  - *--name <container name>* – give container user-defined name
  - *-p <host port>:<container port>* – map host port to container port for access
  - *-it* – indicates interactive on command-line (e.g., for gathering command-line input)
  - *--rm* – container automatically deleted when it exits or stops
  - *-d* – container runs in detached mode (e.g., for continually running web listeners)
- See *docker run --help* for additional info

Container Status

Develop Intelligence

- Use *docker logs <container name>* to see log output from container
- Use *docker logs -f <container name>* for ongoing monitor of log output
- Helpful for troubleshooting issues with container creation, startup, or operation

Container Commands

Develop Intelligence

- *docker exec* can be used to execute a command in a running container
- For example, *docker exec <container name> ls -a* to see container file contents
- *docker exec -it <container name> /bin/bash* for interactive command-line session in container (for Linux-based images that support bash)

Managing Containers

Develop Intelligence

- *docker rm <container name>* or *docker rm <container ID>* removes container
- *-f* argument used to remove a running container
- Can use *docker rm -f $(docker ps -aq)* to remove all (CAUTION)
- *docker ps -aq* lists all containers in quiet mode (returns container ID's only)

Lab: Creating a Docker Container

Develop Intelligence

Data Storage in Docker

Develop Intelligence

- Running containers generate data
- They also need access to persistent data
- We can use the host machine via a “bind mount” – mount a local file or folder into a container
  - Problems: Not easily managed by docker CLI
  - Rely on the host machine having a specific directory structure
- Better is to use a docker construct called a “volume”
  - New directory created in the host machine’s docker storage directory
  - Easier to back up
  - Work the same on both Linux and Windows containers
  - Can be safely shared between containers
  - Content can be pre-populated by the container

Common Docker Issues – How to Identify and Fix

- Dependency issues with base image:
  - `RUN apt-get clean && apt-get update` (clears cache in event base image has been updated in the registry)
- You may need to do this outside the container as well
- Container naming collisions:
  - If you try to use a container name that exists, you'll throw an error – EVEN if the container isn't being used. Remove it to use the name again.

Odds & Ends

- Cheat sheets/documentation for Docker:
  - https://swissarmydevops.com/wp-content/uploads/2020/11/Docker_Cheat_Sheet-1.pdf
  - https://docs.docker.com/
  - https://github.com/wsargent/docker-cheat-sheet

Diving Deeper into Docker and Containers

Develop Intelligence

Lab: Containerize Two Apps Locally

Run through this tutorial twice through: [Run an ASP.NET Core app in a container]

Discussion: Databases in Docker

Develop Intelligence

- Should you even put databases in containers? Why or why not?
- If you are using a *stateful* application in a system made for running stateless applications, you'll have problem
- You can do it – just think about why you want to do so
- How DO we manage state in stateless apps (microservices)

Lab: Containerize a Database Server

https://phoenixnap.com/kb/mysql-docker-container

Code Challenge: Python Web App Dockerfile

Develop Intelligence

Notes: Previous examples used IIS and a .NET Core application. Here, you will build an image for a Python web app that uses the flask web application framework that is hosted by Apache HTTP Server

Docker Compose as a Dev Tool

Develop Intelligence

- Used to run multi-container applications
- Declaratively configures your app's services as a unit
- All services can be started with one command

Docker Compose as a Dev Tool

Develop Intelligence

- May require separate install
- To check for presence, run `*docker-compose --version*`
- To install:

```
#!/bin/bash
VERSION=$(curl --silent https://api.github.com/repos/docker/compose/releases/latest | grep -Po '"tag_name": "\K.*\d')
DESTINATION=/usr/local/bin/docker-compose
sudo curl -L https://github.com/docker/compose/releases/download/${VERSION}/docker-compose-$(uname -s)-$(uname -m) -o $DESTINATION
sudo chmod 755 $DESTINATION
docker-compose --version
```

Docker Compose as a Dev Tool

- docker-compose.yml used to define containers and relationships
- Uses YAML (Yet Another Markup Language)
- General format:

demos > docker-compose > docker-compose.yml

```yaml
container_name:
  property: value
  - or options
```

Docker Compose as a Dev Tool

Develop Intelligence

- Supports all properties available with `*docker run*`
- Uses a `*links*` property to link two containers together
- In it, specify required connections to existing container definition

Docker Compose as a Dev Tool

Develop Intelligence

- Define secondary container(s) using same format
- Linked by identifier
- Multiple containers can be defined together in single YAML file

Docker Compose as a Dev Tool

Develop Intelligence

- `*docker-compose up*` uses YAML file to launch all containers with one command
- Use `*docker-compose up <name>*` to bring up single container
- `*-d*` argument runs in background (similar to use with `*docker run*`)

Docker Compose as a Dev Tool

Develop Intelligence

- `*docker-compose ps*` displays details for all launched containers
- `*docker-compose logs*` display all logs for multi-container “unit”
- `*docker-compose scale web=#*` scales the number of web containers (use 1 to scale back down)

Docker Compose as a Dev Tool

Develop Intelligence

- `*docker-compose stop*` to stop all containers
- `*docker-compose down*` or `*docker-compose rm*` to remove all containers

See https://docs.docker.com/compose/compose-file/

Demo: Using Docker Compose

Develop Intelligence

Lab: Using Docker Compose

Develop Intelligence

Application Bootstrapping with Docker and k8s

Develop Intelligence

- Kubernetes provides a hosting environment for containerized applications
- Once you have a Docker image, you can work entirely within Kubernetes to deploy your app

Open Container Initiative (OCI)

Develop Intelligence

- The OCI is an open governance structure for the express purpose of creating open industry standards around container formats and runtimes
- Docker started it
- They have two specs: runtime-spec and image-spec
- This deals with containers in the abstract

Competing Container Runtimes

Develop Intelligence

- rkt from CoreOS
- Mesos from Apache
- LXC Linux containers

Multi-stage Container Builds

- By setting a dependency in your Docker Compose file, you can make your multi-container app spin up in a specific order.

Lab: Multi-Stage Container Build

Develop Intelligence

Kubernetes and Container Orchestration

Develop Intelligence

Kubernetes (k8s) Overview

Develop Intelligence

- Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services

Demo: k8s in Action on Local Machine

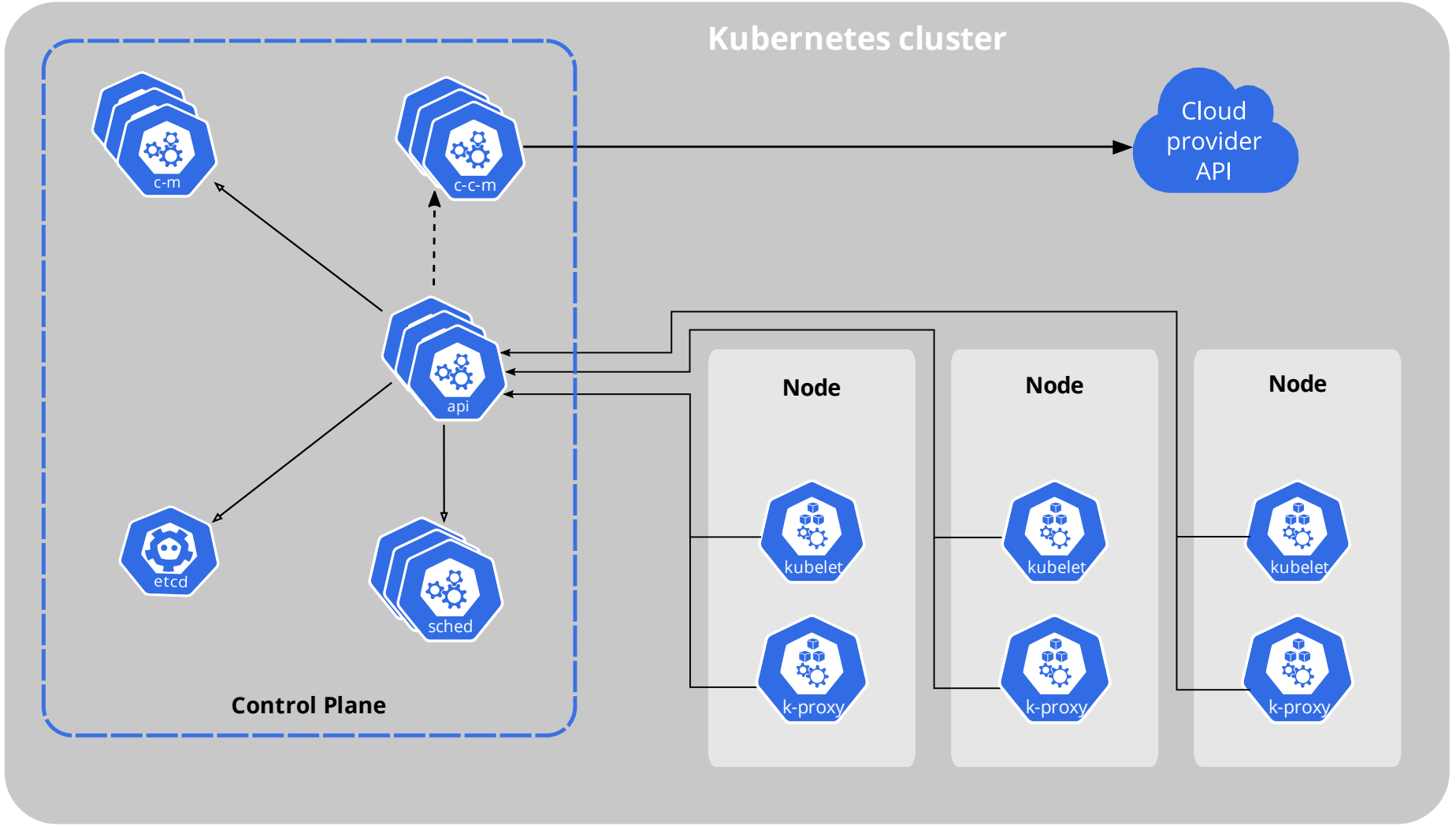
Develop Intelligence

What is an Orchestrator and Why Do We Need It?

Develop Intelligence

- What would it look like to manually control the containers needed for your application as your app scales or containers fail?
- “Orchestration” is the execution of a defined workflow
- We can use an orchestrator to start and stop containers automatically based on your set rules
- What does this open up for you?

Architecture of k8s System



API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

kubelet

kube-proxy

Scheduler

Control plane

Node

Core Components of k8s

Develop Intelligence

- When you deploy Kubernetes, you get a cluster.
- A Kubernetes cluster consists of a set of worker machines, called nodes, that run containerized applications. Every cluster has at least one worker node.
- The worker node(s) host the Pods that are the components of the application workload.
- The control plane manages the worker nodes and the Pods in the cluster.

k8s Resource/Manifest

Develop Intelligence

- Kubernetes manifests are text files, usually written in YAML
- They are used to create, modify and delete Kubernetes resources

Lab: Set up Local k8s Dev Environment

Activate Kubernetes now, via Docker Desktop

Use kubectl from command line to see k8s config data

Kubernetes Architecture

- Kubernetes automates and monitors the lifecycle of a stateless application
- It can scale the app up or down and ensure it keeps running
- Kubernetes cluster consists of computers called nodes
- The basic unit of work in kubernetes is called a pod, which is a group of one or more containers
- Kubernetes can be divided into planes
  - Control plane - the actual pods that the kubernetes core components runs on, exposing the api, etc...
  - Data Plane - everything else meaning the nodes and pods which the application runs on

Kubernetes Architecture

Develop Intelligence

- In short, control plane is what monitors the cluster, schedules the work, makes the changes, etc…
- The nodes are what actually do the real work, report back to the master, and watch for changes

Kubernetes Control Plane & Data Plane

Develop Intelligence

- Kubernetes is actually a collection of pods implementing the k8s api and the cluster orchestration logic (AKA the Kubernetes control plane)
- The application/data plane is everything else, meaning all the nodes that host all the pods that the application runs on
- The *controllers* of the control plane implement control loops that repeatedly compare the desired state of the cluster to its actual state
- When the state of the cluster changes, controllers take action to bring it back inline with desired state

Declarative and Desired State

Develop Intelligence

- Kubernetes supports a declarative model – we can send the api a yaml file in a declarative form
- Desired state means that we specify in the yaml file our desired state and the cluster takes responsibility to make sure it will happen
- We describe the desired state using a yaml or json file that serves as a record of intent, but we do not specify how to get there (this is kubernetes responsibility to get us there)
- Things could change or go wrong over the lifetime of the cluster (node failing, etc…), Kubernetes is responsible to always make sure that the desired state is kept intact
- Kubernetes control plane controllers are always running in a loop and checking that the actual state of the cluster matches the desired state, so that if any error occurs they kick in and rectify the cluster

Reconciling state

Develop Intelligence

- Watch for the spec fields in the YAML files
- The *spec* describes *what we want the thing to be*
- Kubernetes will *reconcile* the current state with the spec (technically, this is done by a number of *controllers*)
- When we want to change a resource, we update the *spec*, and reapply
- Kubernetes will then *converge* that resource

k8s Pods: The Basic Building Block

Develop Intelligence

- A pod represents a set of running containers in your cluster
- Worker nodes host pods
- The control plane manages the worker nodes and the pods inside them

k8s Pods: The Basic Building Block

Develop Intelligence

- Pod is the basic execution and scaling unit in Kubernetes
- Kubernetes runs containers but always inside pods
- It is like a sandbox in which containers run (abstraction)
- A pod is a group of containers:
  - Running together (on the same node)
  - Sharing resources (RAM, CPU; but, also, network, volumes, etc…)
- A Pod models an application-specific “logical host”

Pod state

Develop Intelligence

- Pods are considered to be relatively ephemeral (rather than durable) entities
- Pods do not hold state, so if a pod crashes, Kubernetes will replace it with another
- Multiple instances of the same pod are called replicas

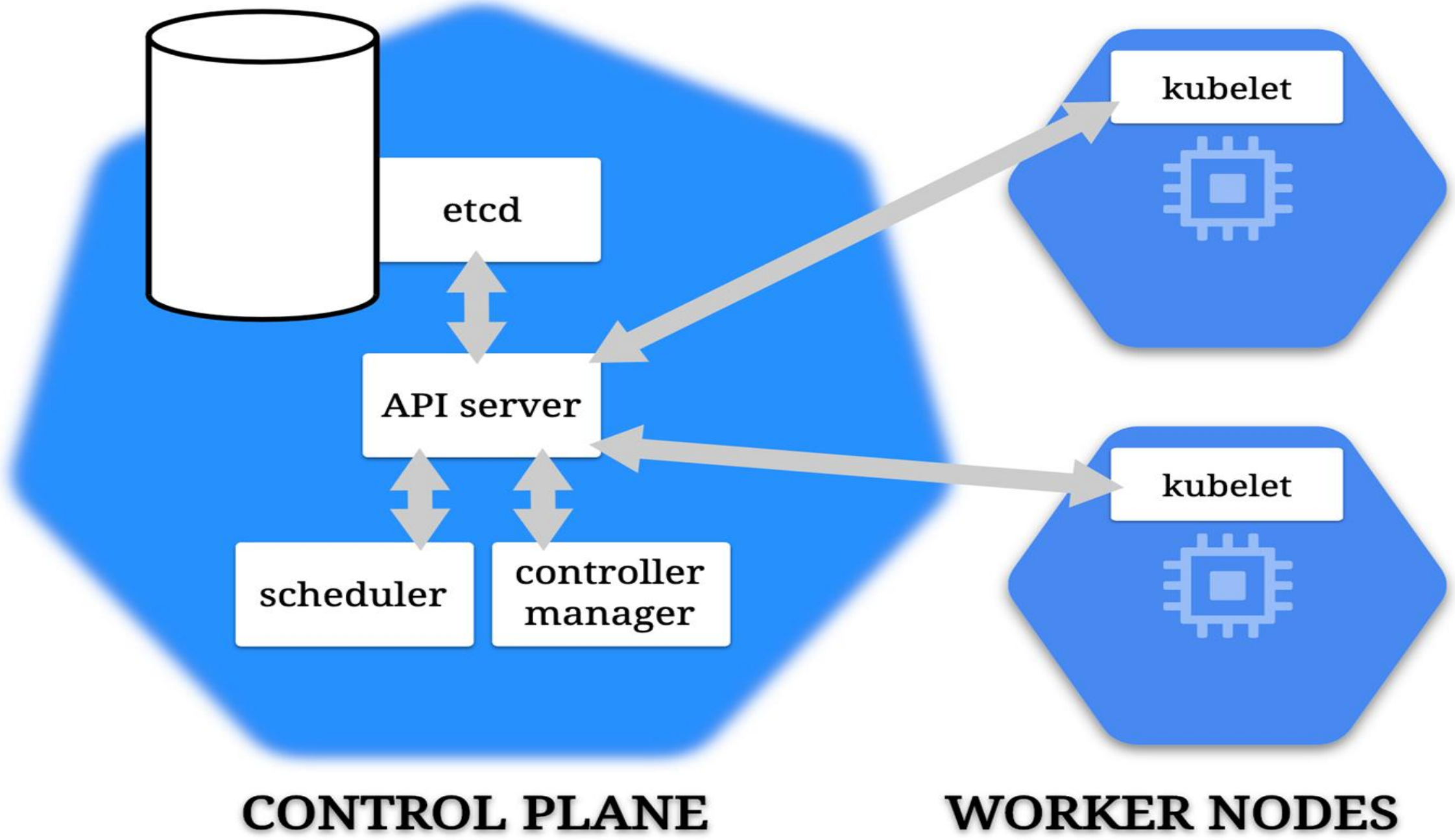
Deployments

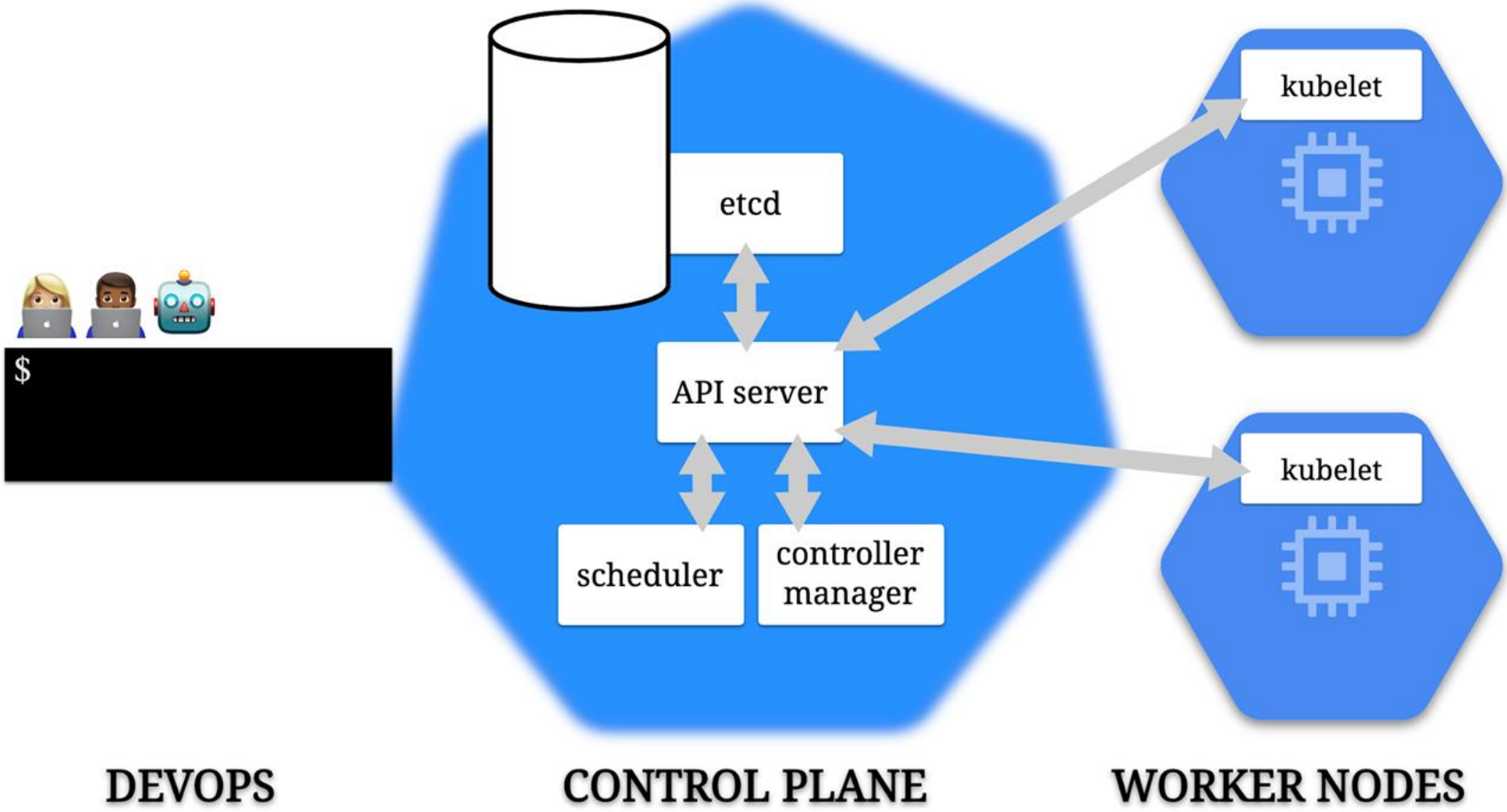
Develop Intelligence

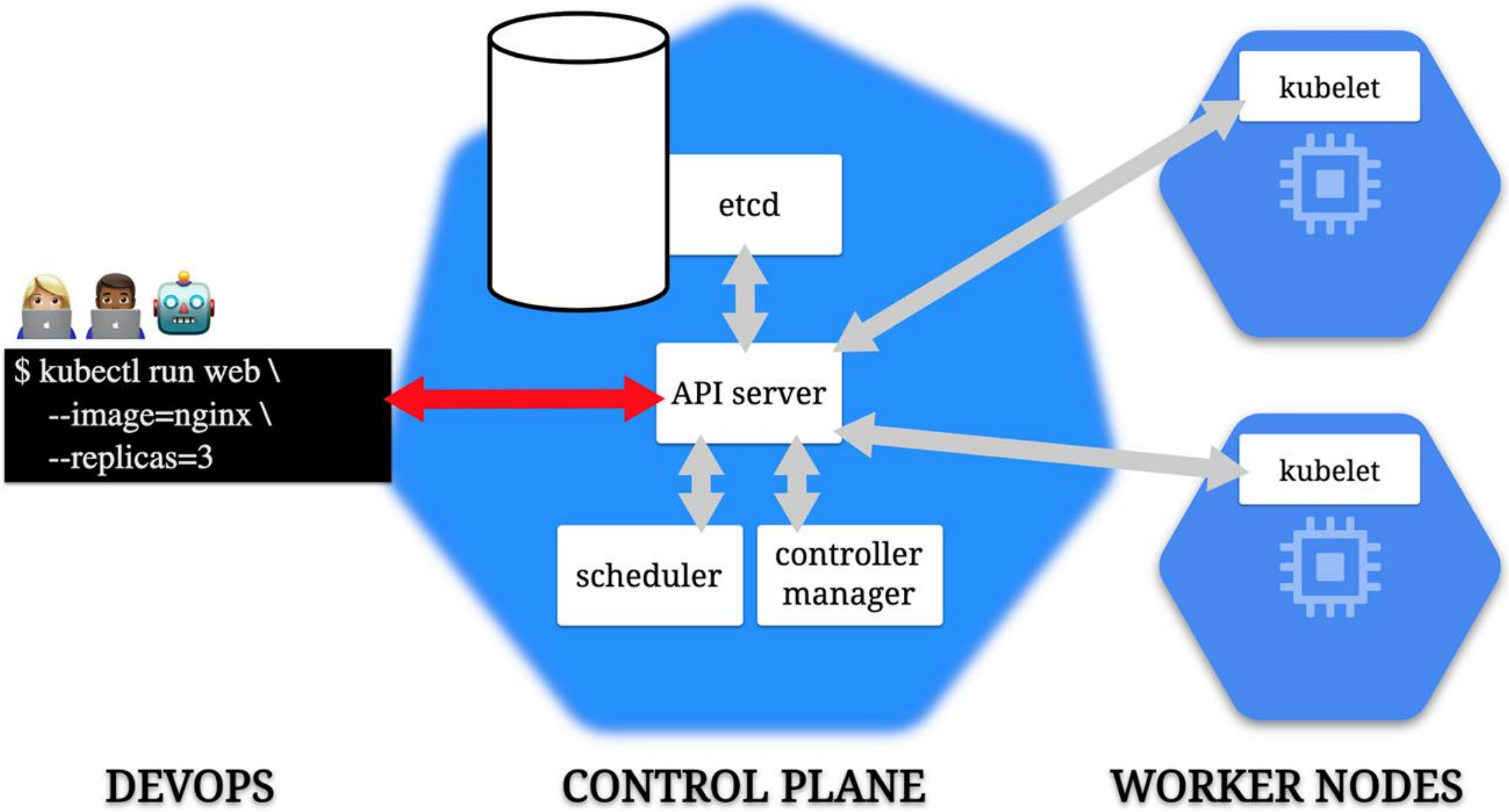
- A *Deployment* controller provides declarative updates for Pods and ReplicaSets
- The Deployment Object gives us a better way of handling the scaling of pods
- The advantage of using Deployment versus using a replicaset is having rolling updates support for the pod container versions out-of-the-box

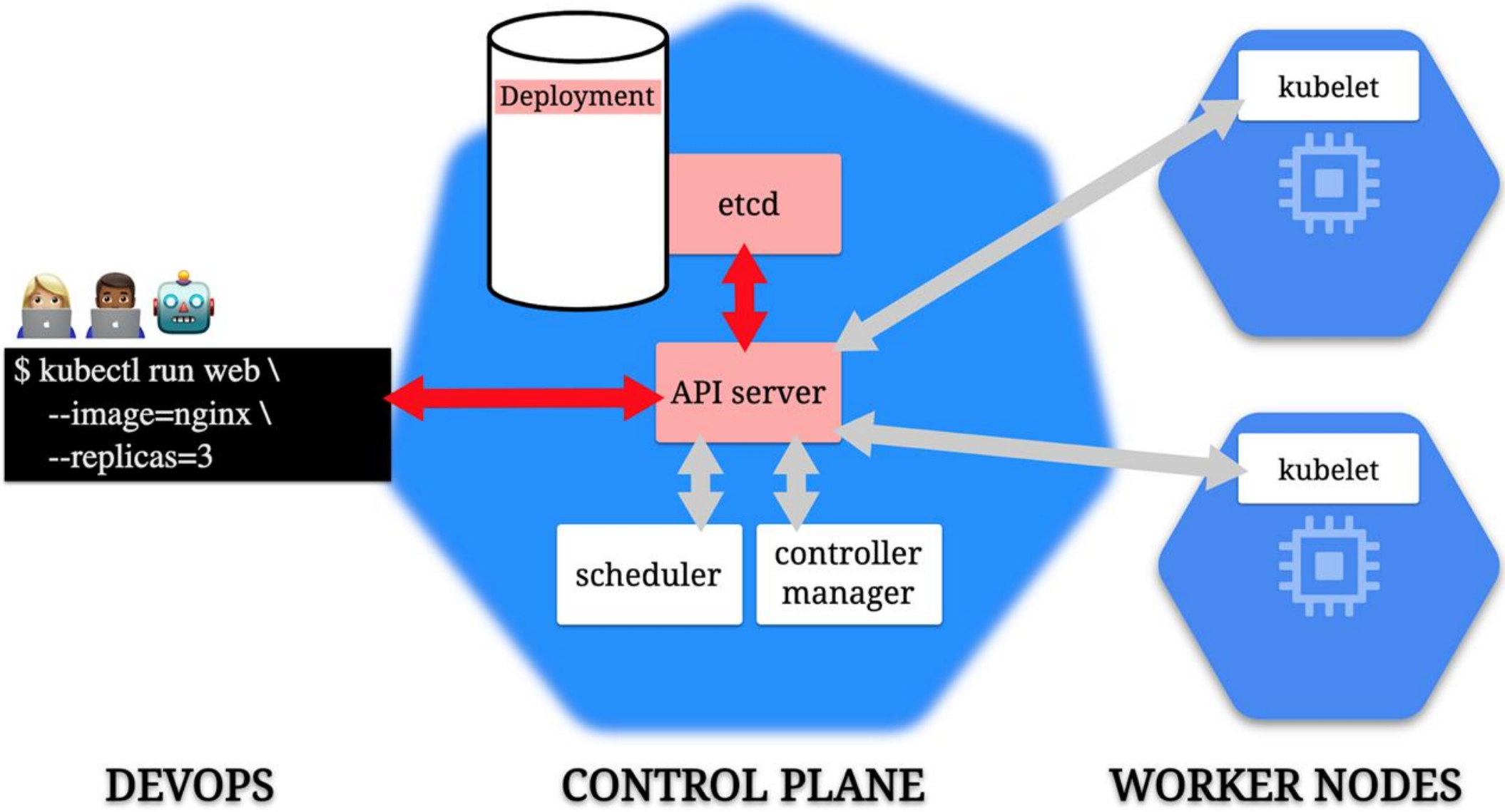
Deployments

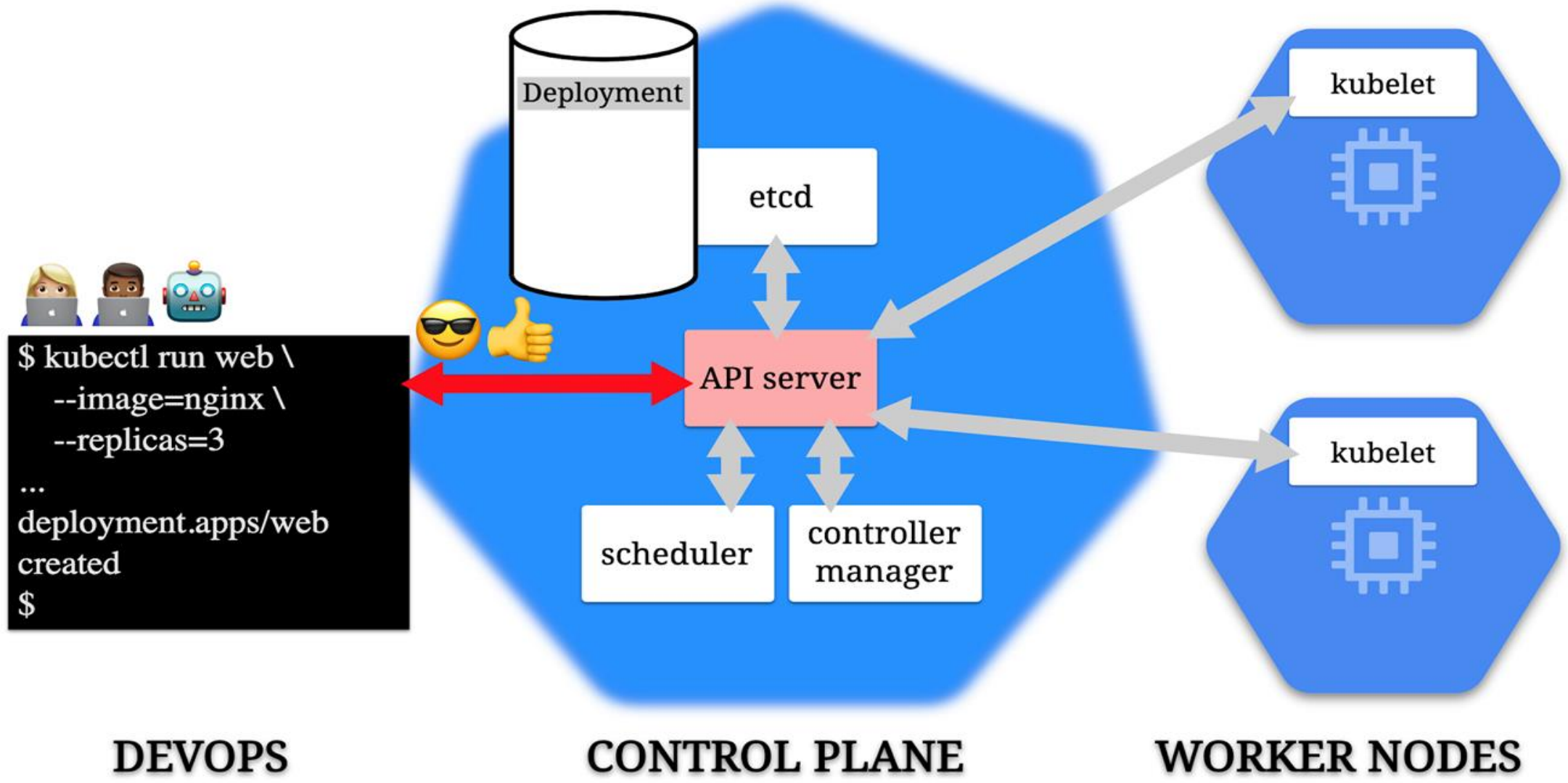
- Every time the application code changes, a new version of the application container is built, and then there is a need to update the Deployment manifest with the new version and tell K8s to apply the changes
- K8s will then handle the rolling-out of this newer version, terminating pods with the old version as it spins up the new pods with the updated container
- This means that at some point we will have multiple versions of the same application running at the same time

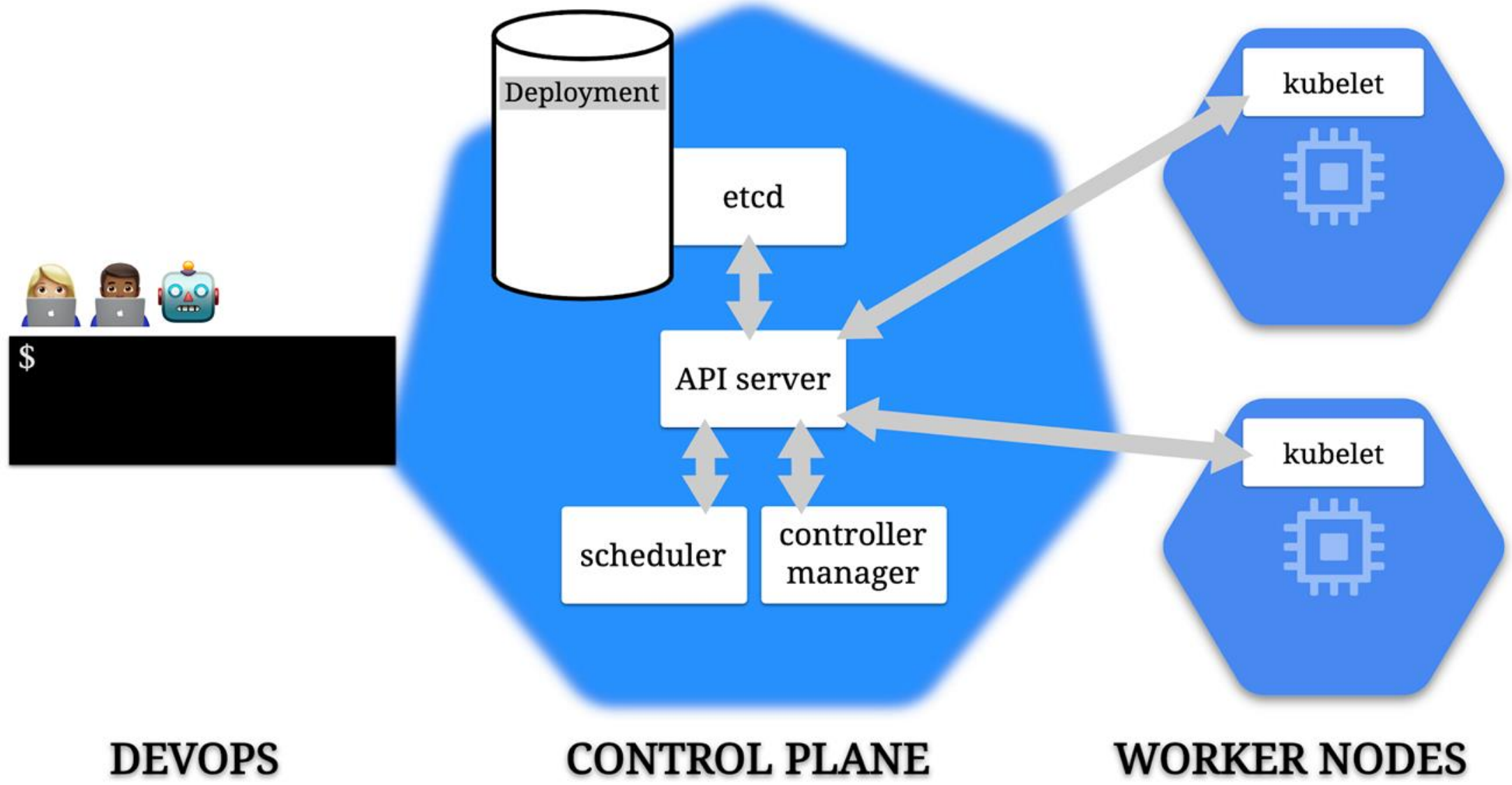

etcd
API server
scheduler
controller manager
kubelet
kubelet
CONTROL PLANE
WORKER NODES

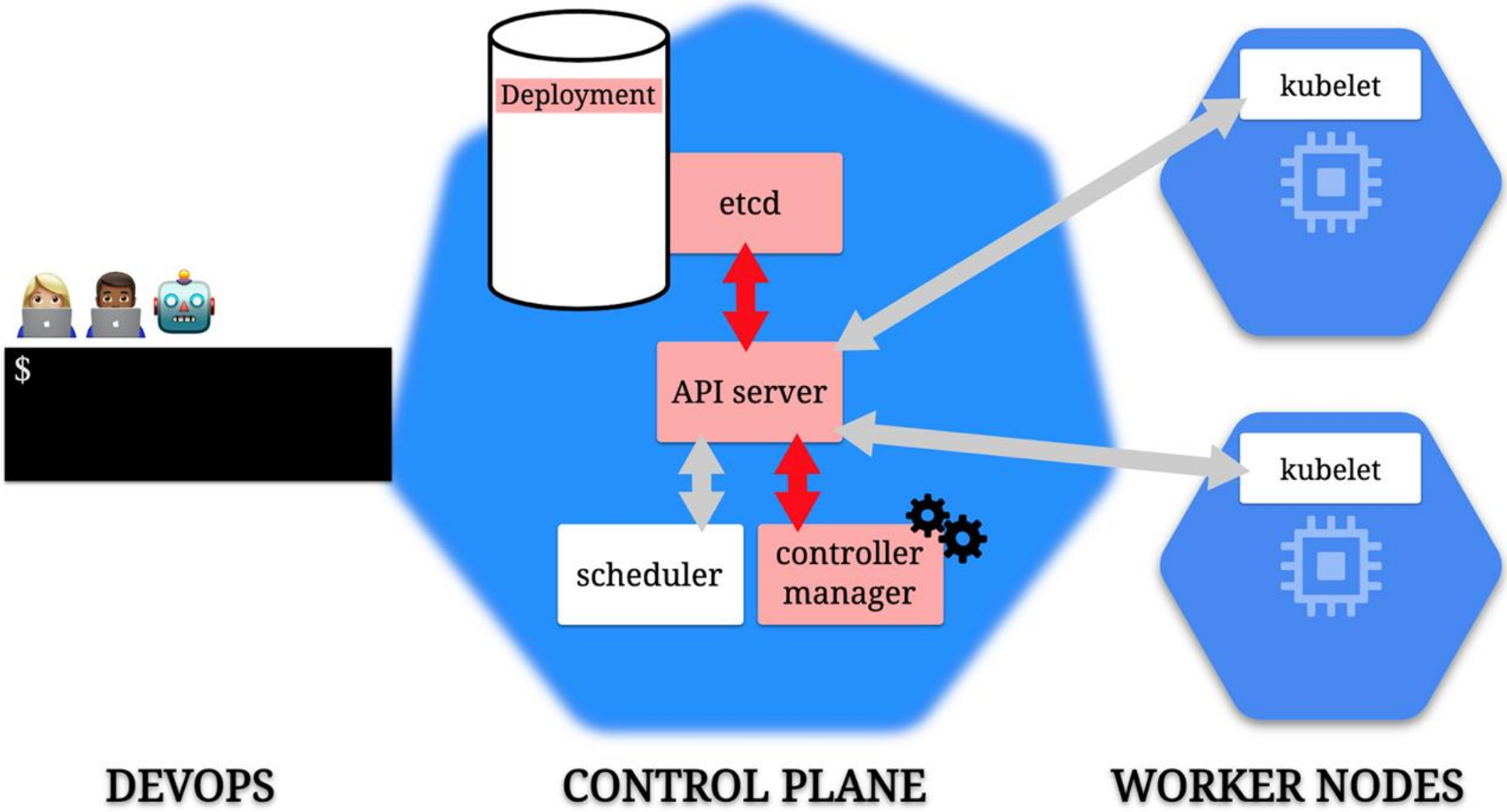

etcd
API server
scheduler
controller manager
kubelet
kubelet
$
DEVOPS
CONTROL PLANE
WORKER NODES

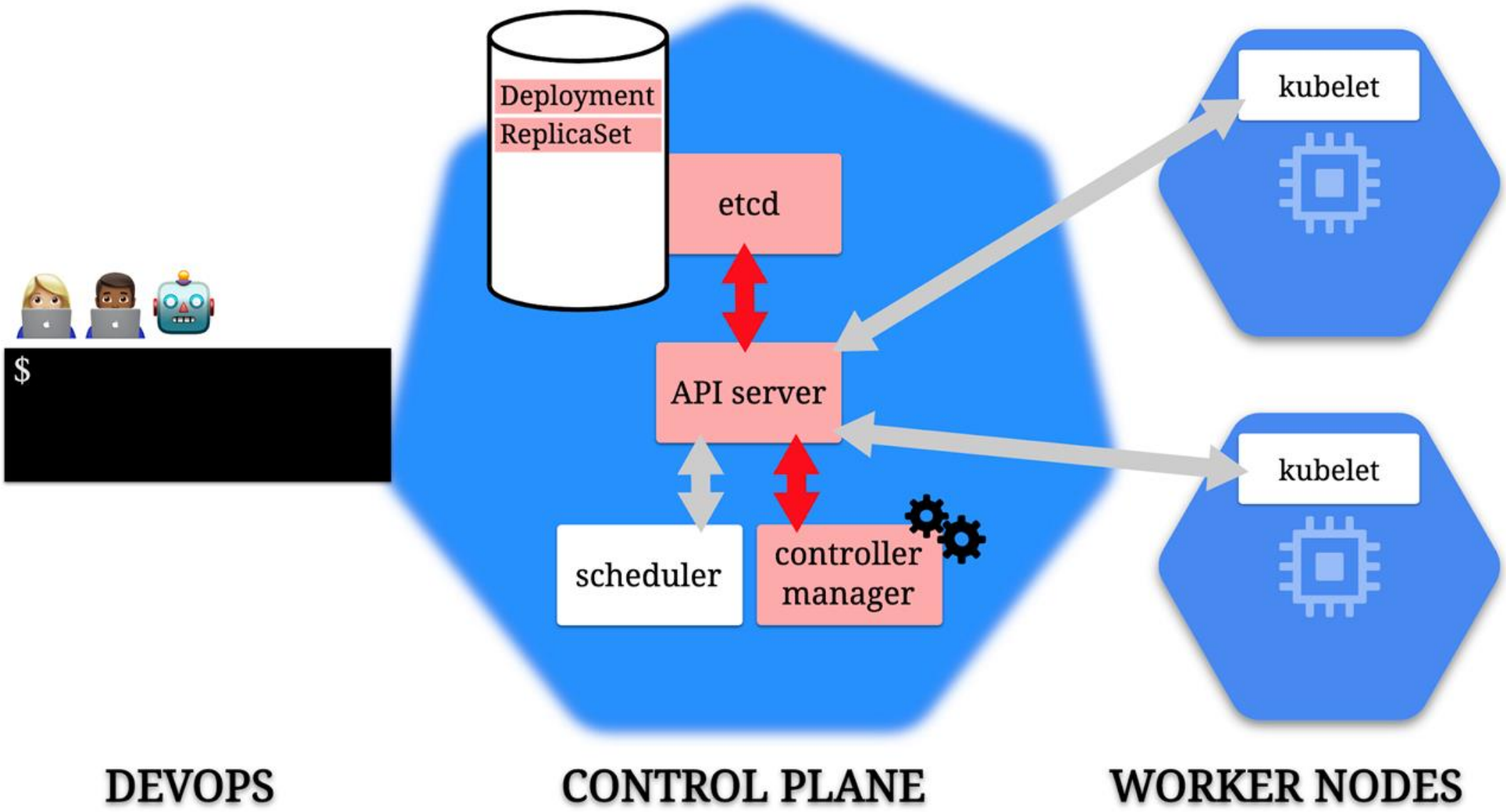

etcd
kubelet
$ kubectl run web \
--image=nginx \
--replicas=3
API server
kubelet
scheduler
controller
manager
DEVOPS
CONTROL PLANE
WORKER NODES

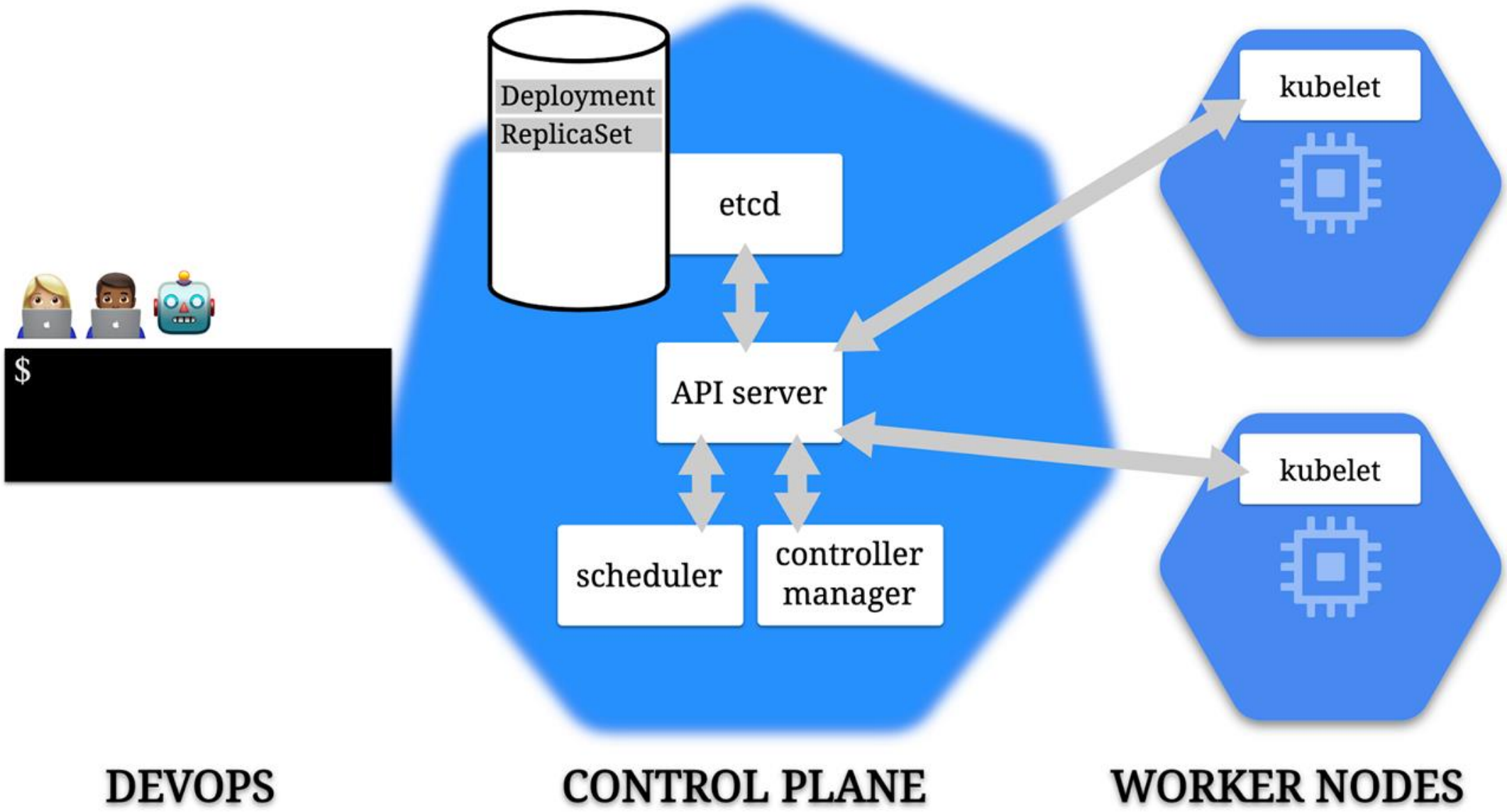

Deployment
etcd
kubelet
$ kubectl run web \
--image=nginx \
--replicas=3
API server
kubelet
scheduler
controller
manager
DEVOPS
CONTROL PLANE
WORKER NODES

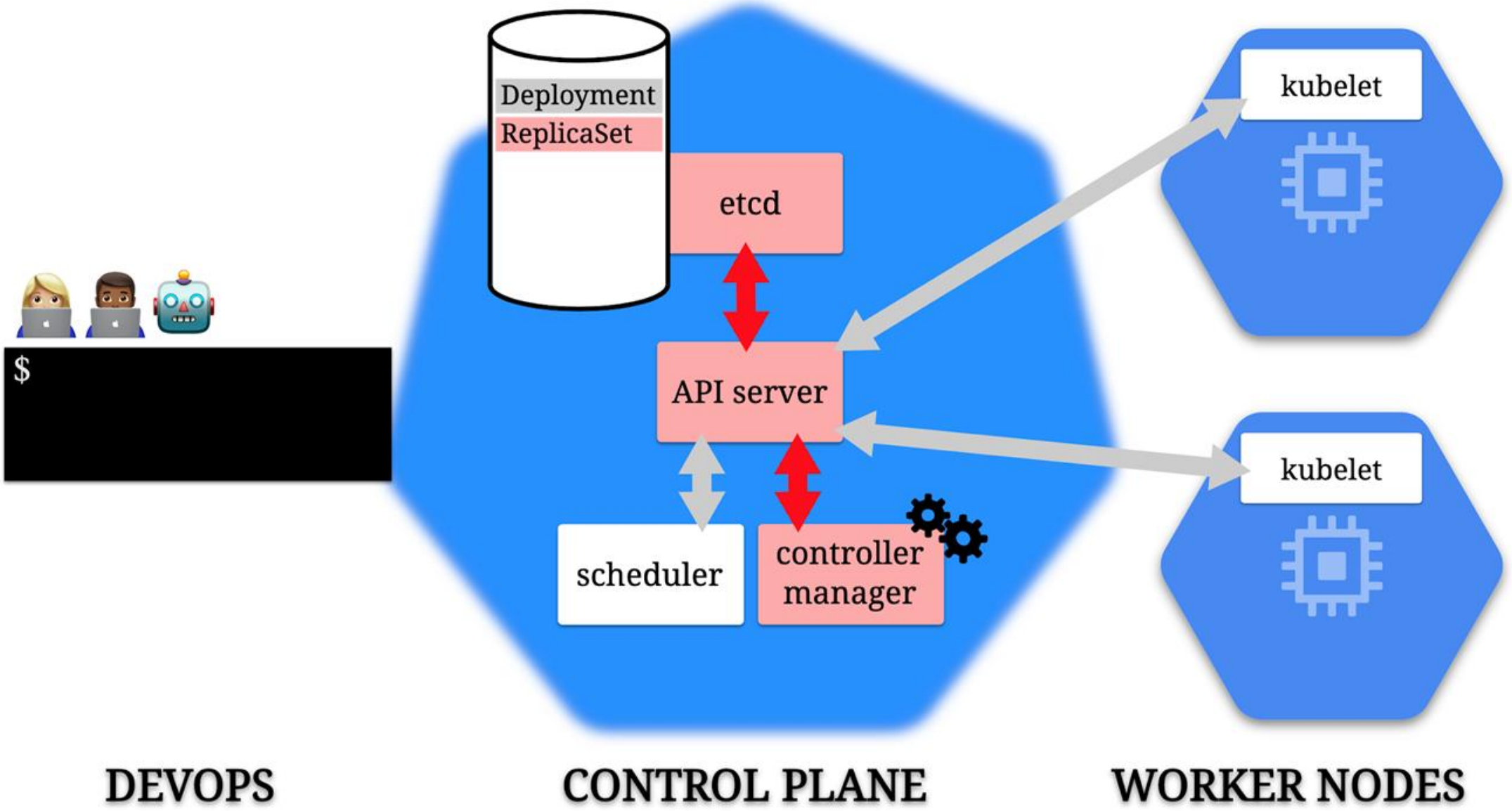

Deployment
etcd
API server
scheduler
controller manager
kubelet
kubelet
$ kubectl run web \
--image=nginx \
--replicas=3
...
deployment.apps/web
created
$
DEVOPS
CONTROL PLANE
WORKER NODES

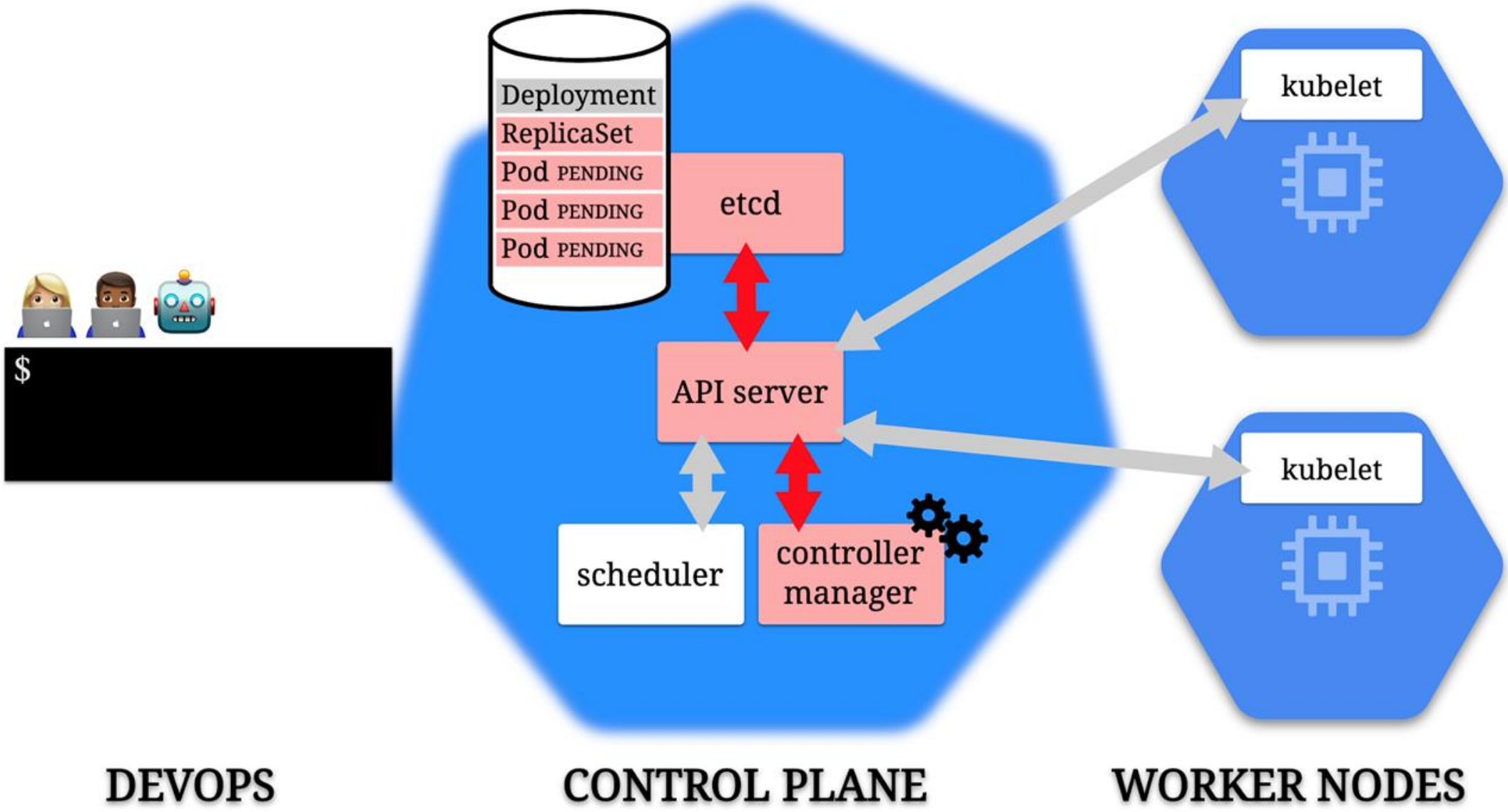

Deployment
etcd
API server
scheduler
controller manager
kubelet
kubelet
$
DEVOPS
CONTROL PLANE
WORKER NODES

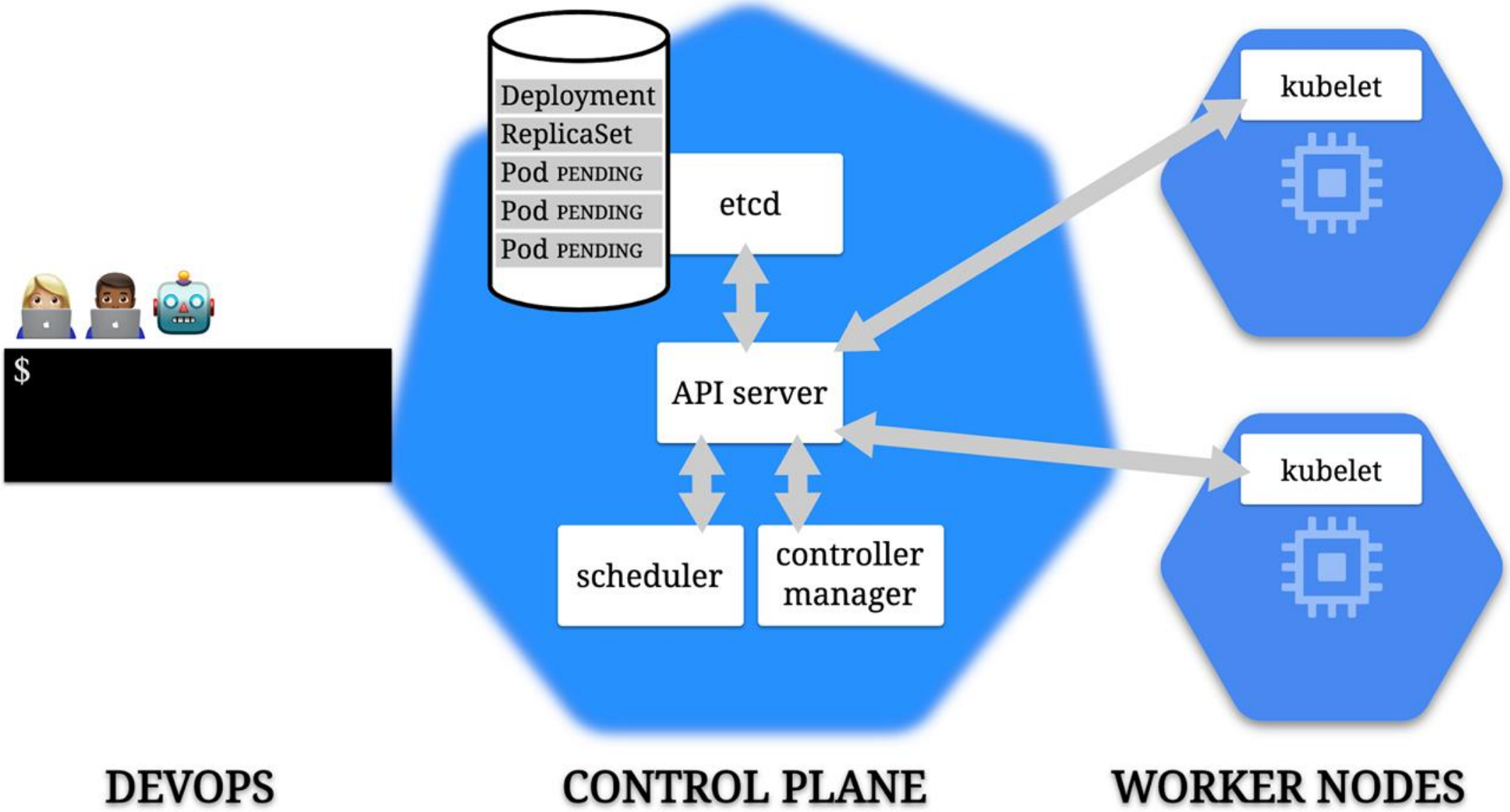

Deployment
etcd
API server
scheduler
controller manager
kubelet
kubelet
$
DEVOPS
CONTROL PLANE
WORKER NODES

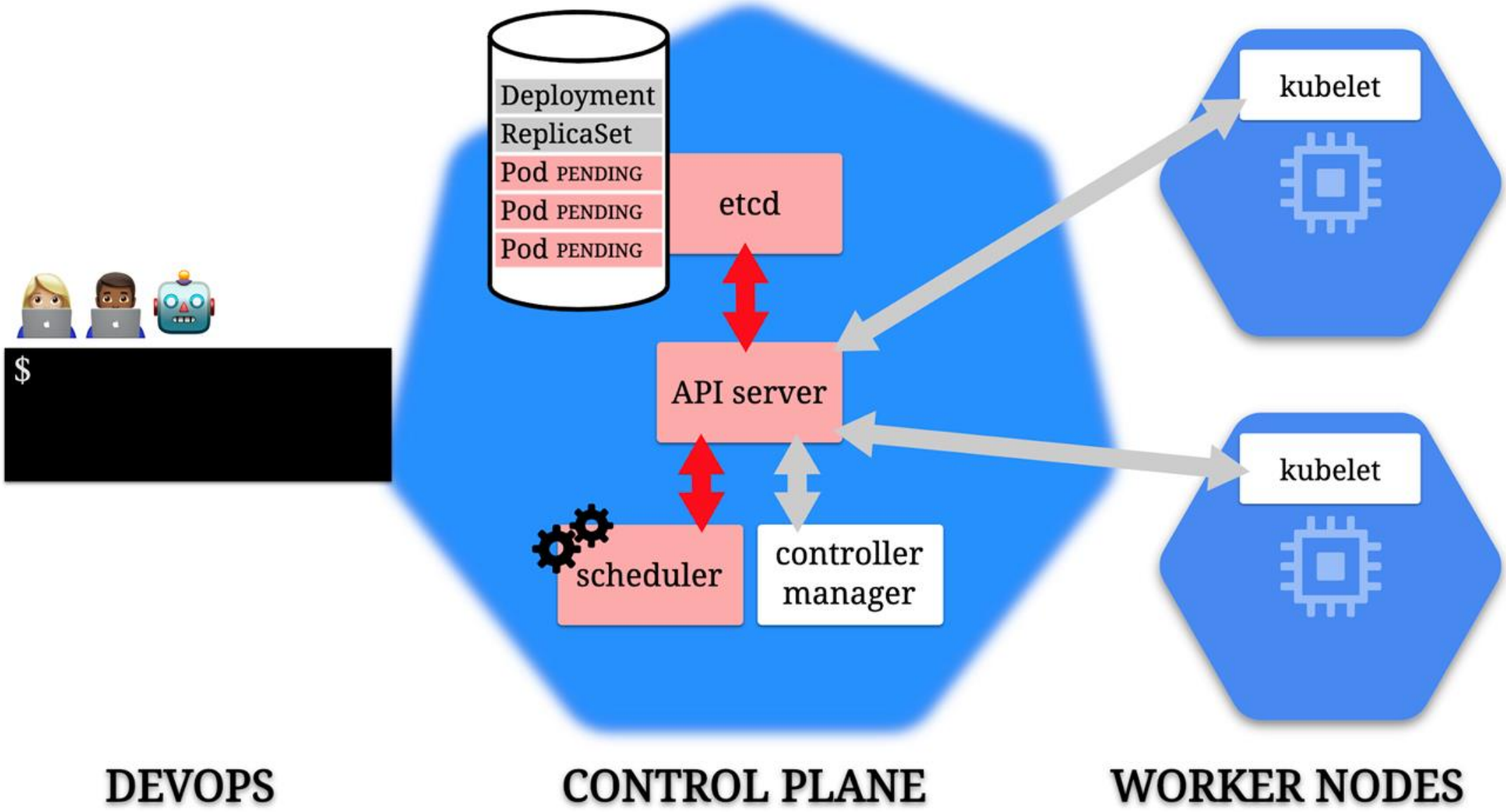

Deployment
ReplicaSet
etcd
API server
scheduler
controller manager
kubelet
kubelet
$
DEVOPS
CONTROL PLANE
WORKER NODES

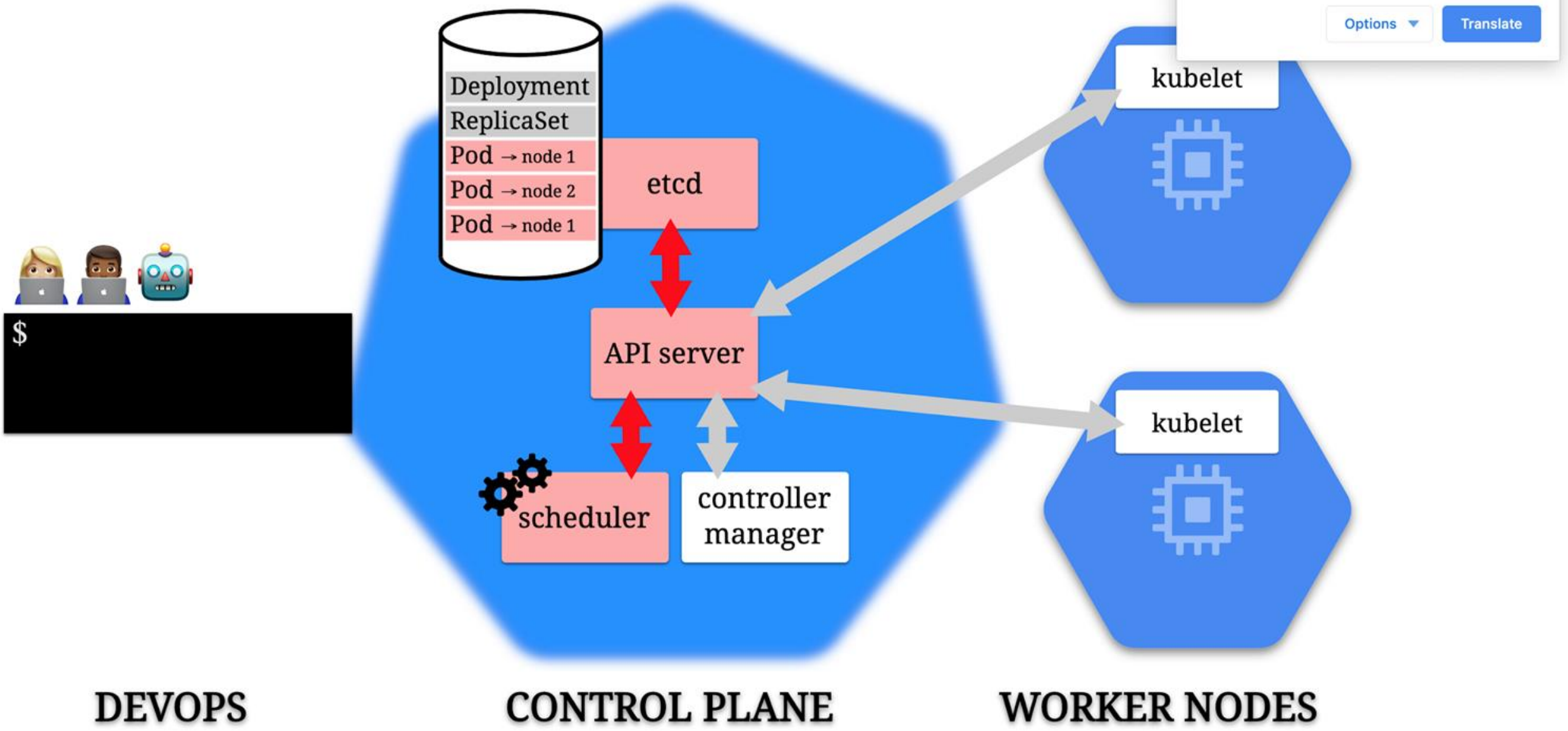

Deployment
ReplicaSet
etcd
API server
scheduler
controller manager
kubelet
kubelet
$
DEVOPS
CONTROL PLANE
WORKER NODES

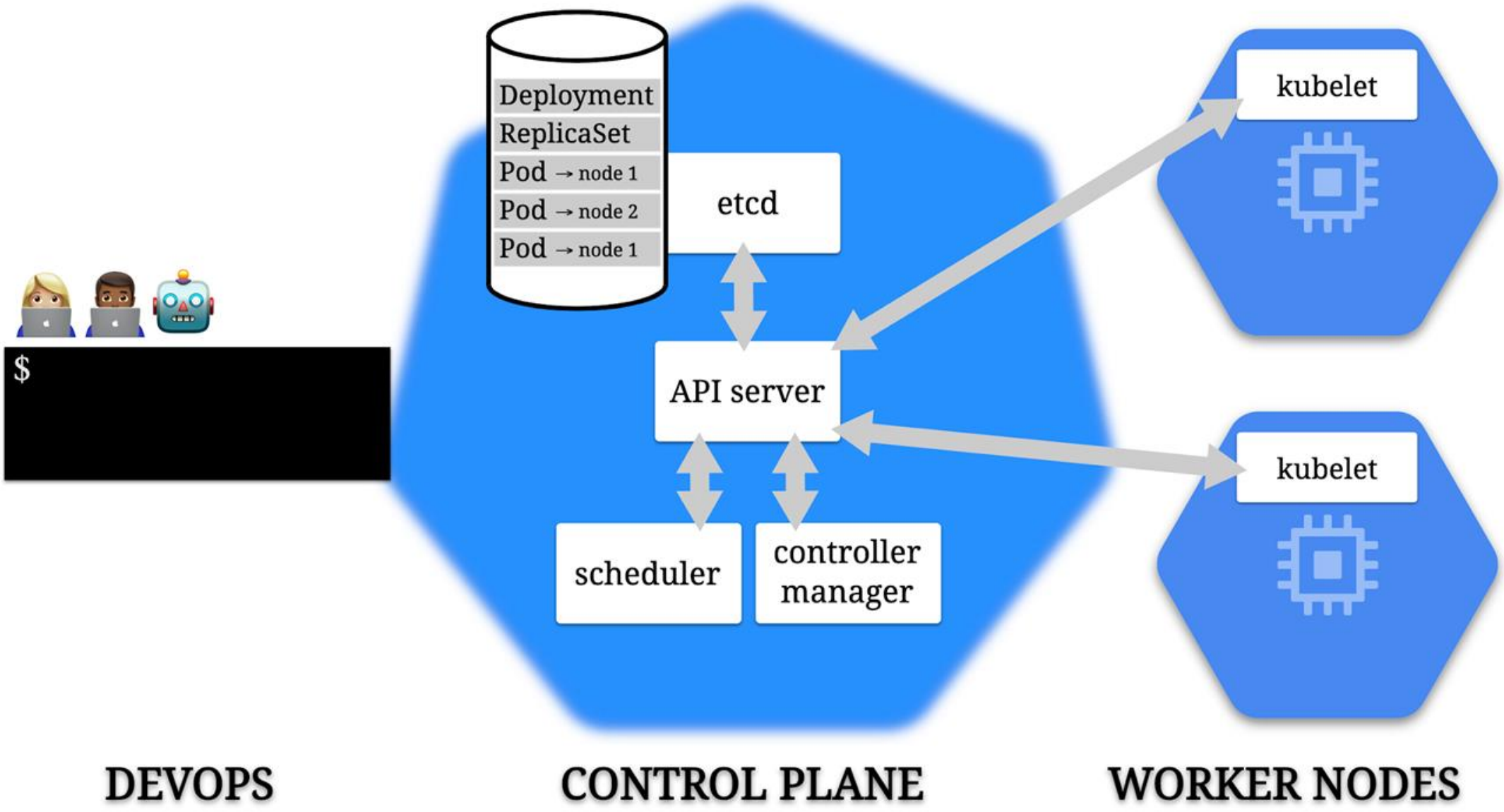

Deployment
ReplicaSet
etcd
API server
scheduler
controller manager
kubelet
kubelet
$
DEVOPS
CONTROL PLANE
WORKER NODES

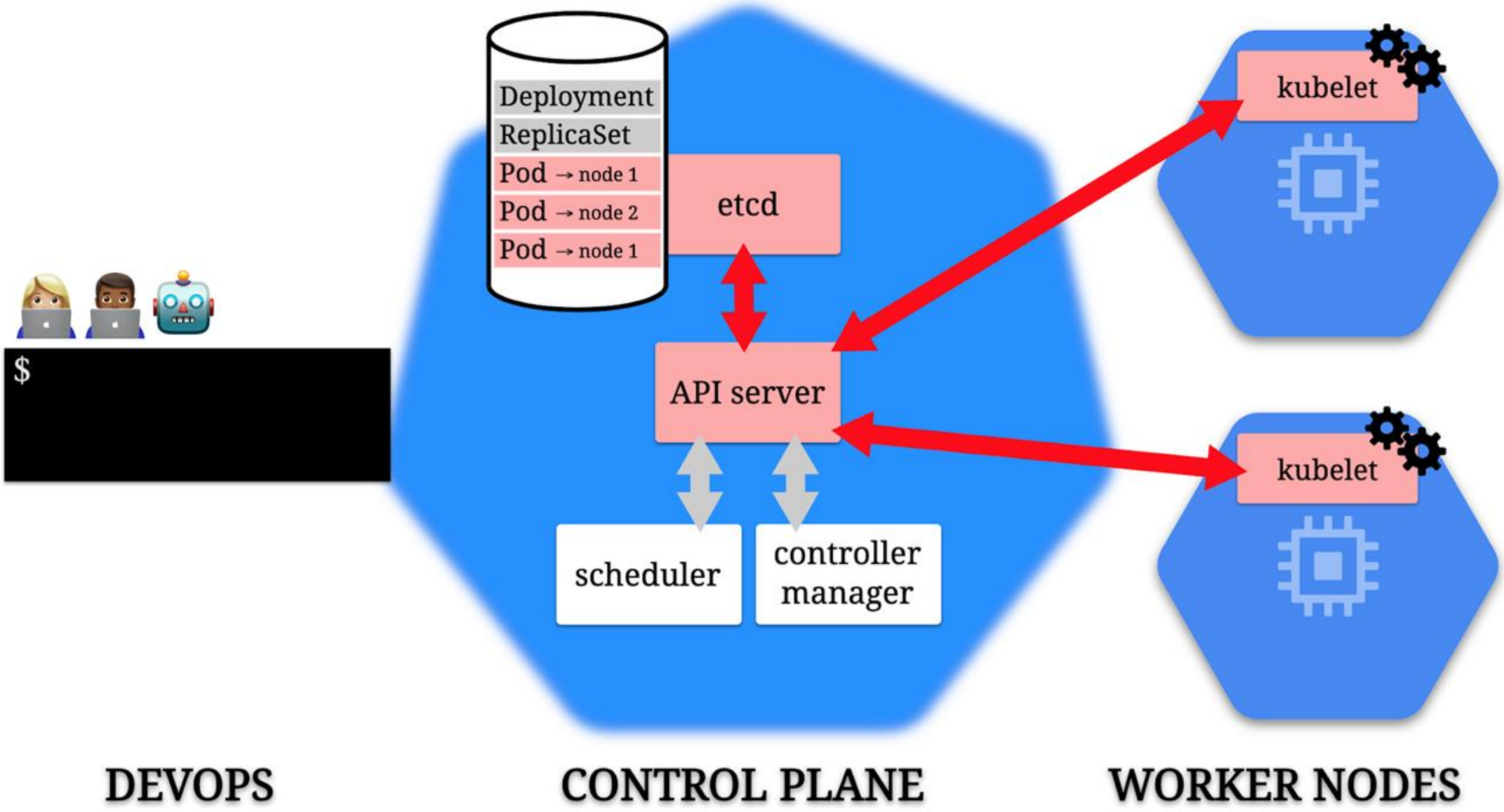

Deployment
ReplicaSet
Pod PENDING
Pod PENDING
Pod PENDING
etcd
API server
scheduler
controller manager
kubelet
kubelet
$
DEVOPS
CONTROL PLANE
WORKER NODES

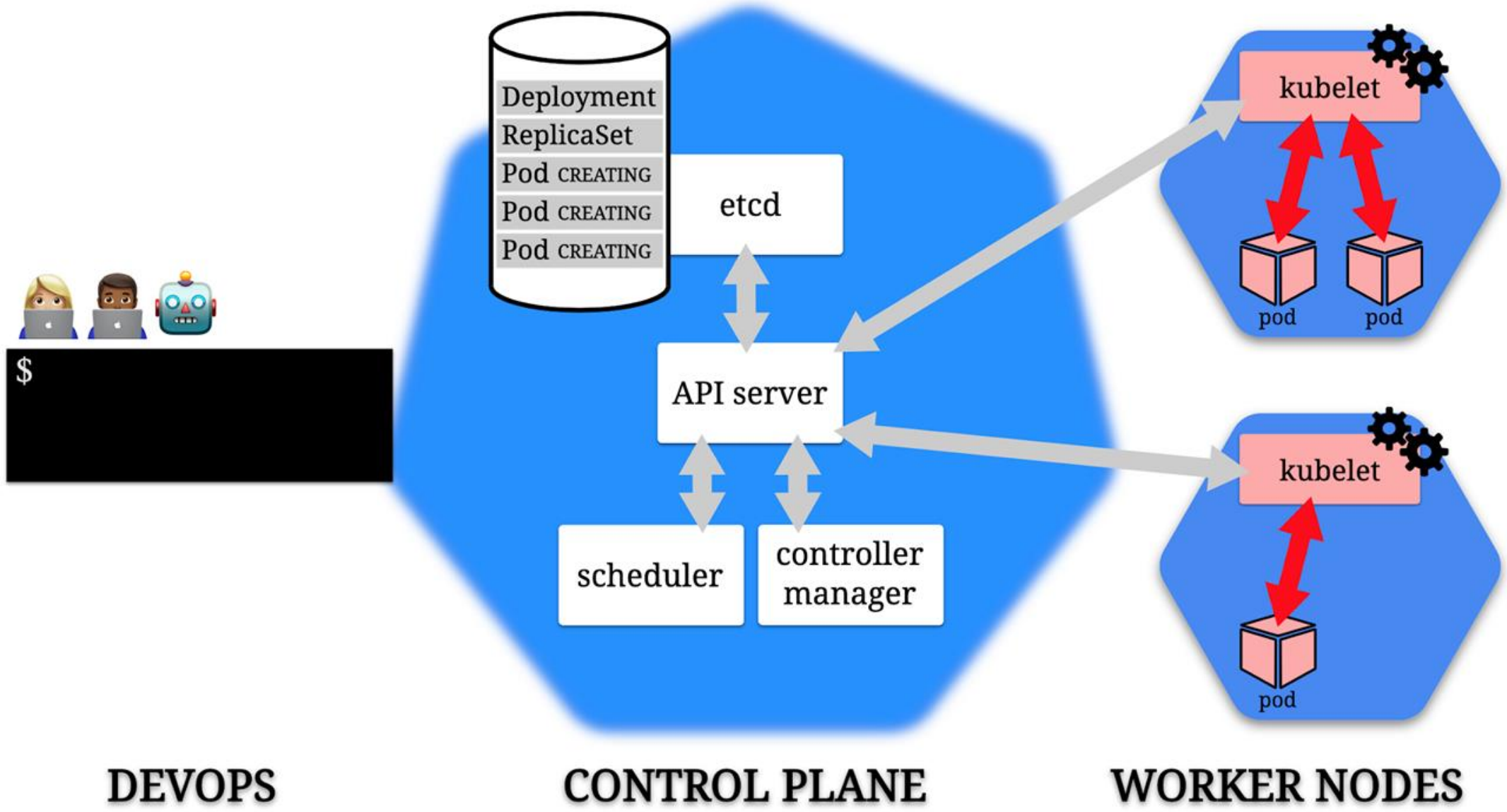

Deployment
ReplicaSet
Pod PENDING
Pod PENDING
Pod PENDING
etcd
API server
scheduler
controller manager
kubelet
kubelet
$
DEVOPS
CONTROL PLANE
WORKER NODES

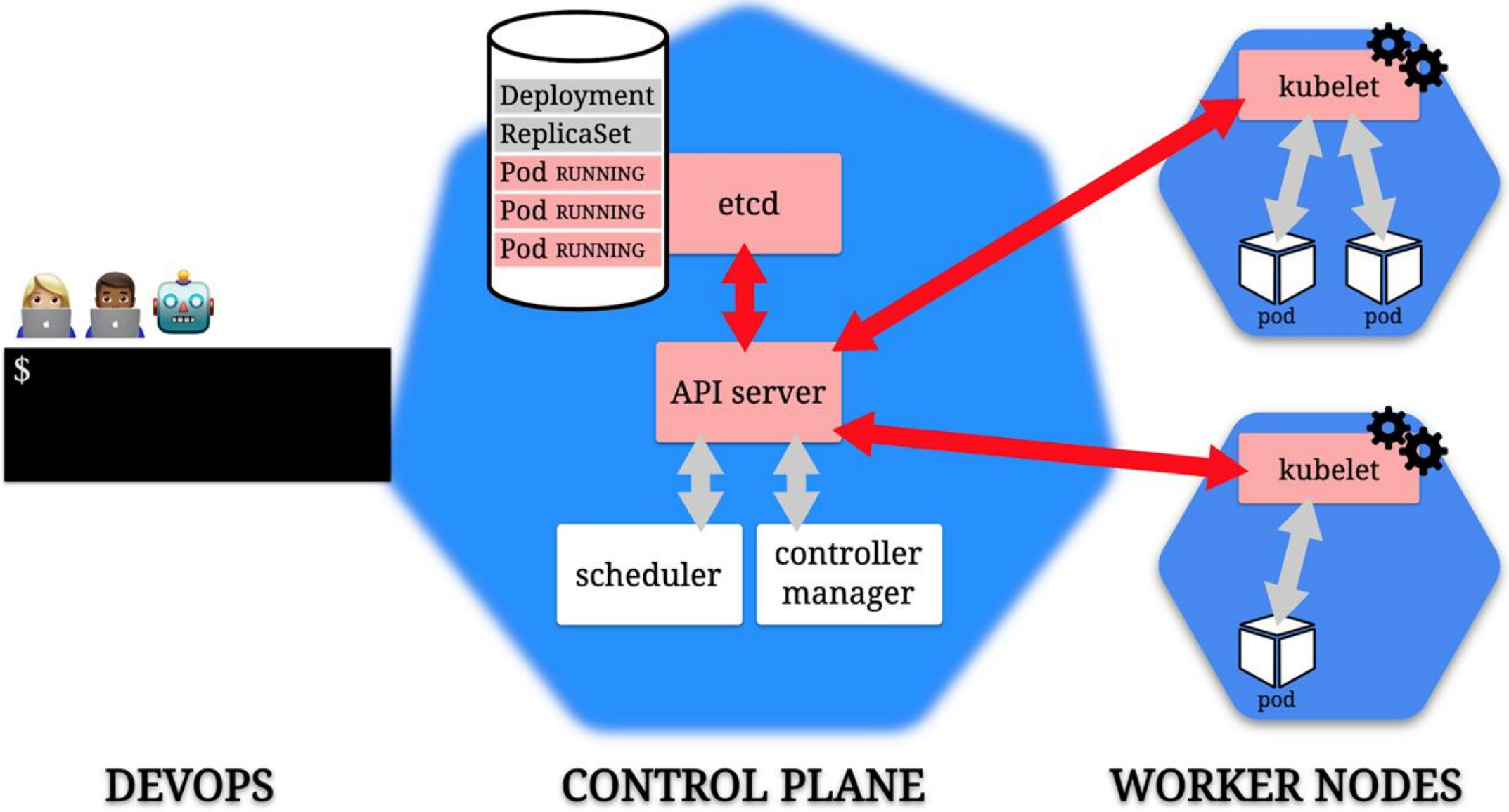

Deployment
ReplicaSet
Pod PENDING
Pod PENDING
Pod PENDING
etcd
API server
scheduler
controller manager
kubelet
kubelet
$
DEVOPS
CONTROL PLANE
WORKER NODES

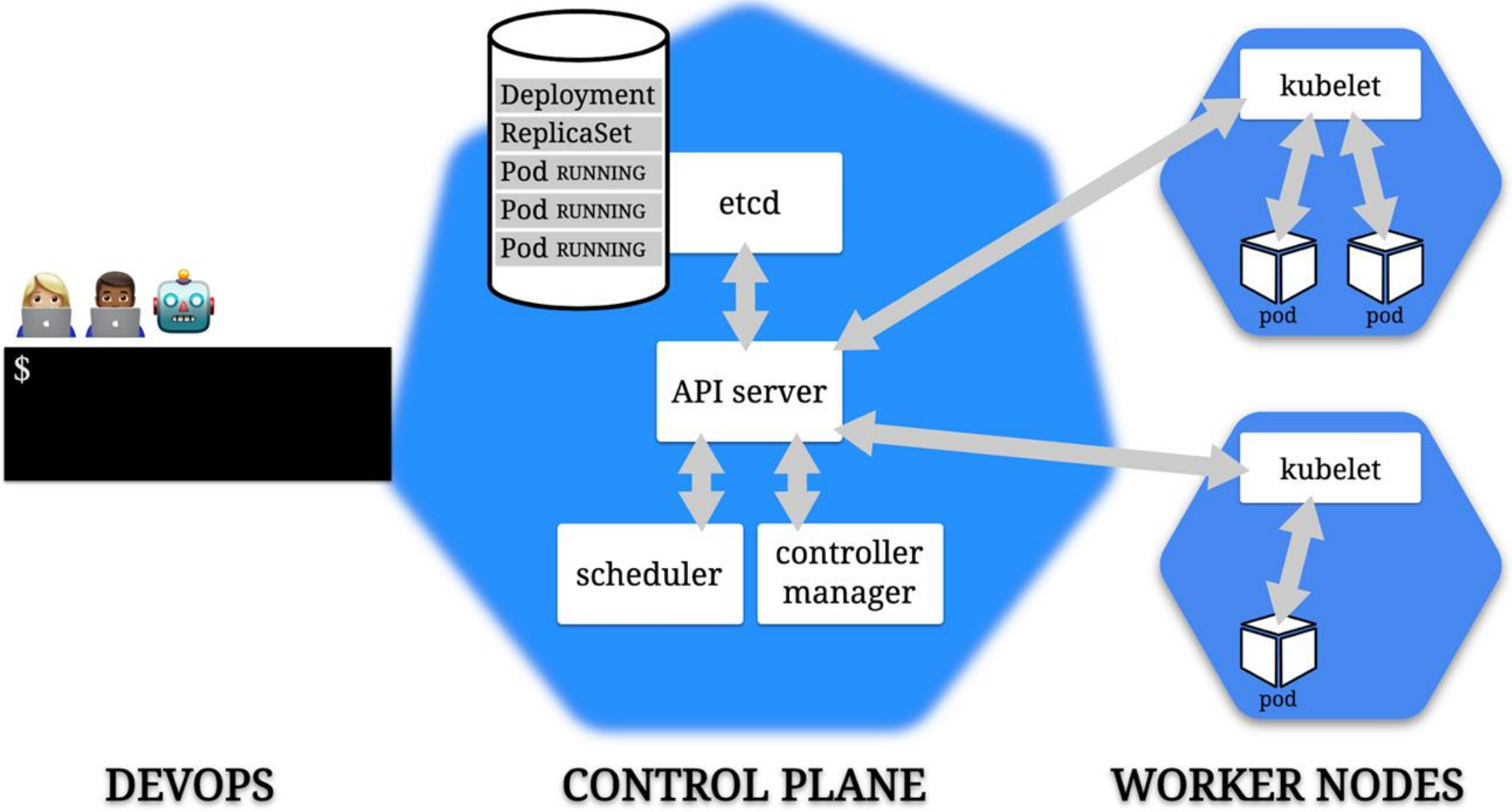

Options
Translate
Deployment
ReplicaSet
Pod → node 1
Pod → node 2
Pod → node 1
etcd
API server
scheduler
controller manager
kubelet
kubelet
$
DEVOPS
CONTROL PLANE
WORKER NODES


Deployment
ReplicaSet
Pod → node 1
Pod → node 2
Pod → node 1
etcd
API server
scheduler
controller manager
kubelet
kubelet
$
DEVOPS
CONTROL PLANE
WORKER NODES


Deployment
ReplicaSet
Pod → node 1
Pod → node 2
Pod → node 1
etcd
API server
scheduler
controller manager
kubelet
kubelet
$
DEVOPS
CONTROL PLANE
WORKER NODES


Deployment
ReplicaSet
Pod CREATING
Pod CREATING
Pod CREATING
etcd
API server
scheduler
controller manager
kubelet
pod
pod
kubelet
pod
$
DEVOPS
CONTROL PLANE
WORKER NODES


Deployment
ReplicaSet
Pod RUNNING
Pod RUNNING
Pod RUNNING
etcd
API server
scheduler
controller manager
kubelet
pod
pod
kubelet
pod
$
DEVOPS
CONTROL PLANE
WORKER NODES


Deployment
ReplicaSet
Pod RUNNING
Pod RUNNING
Pod RUNNING
etcd
API server
scheduler
controller manager
kubelet
pod
pod
kubelet
pod
$
DEVOPS
CONTROL PLANE
WORKER NODES

Control Plane: The API

Develop Intelligence

- The fundamental fabric of Kubernetes
- RESTful API
- Enables all communication between k8s components, and all operations they can do
- Enables external user commands into the cluster

Master in more detail

- The Master is composed of multiple components
- **kube-apiserver** which is the front facing interface into the master exposes its interface via REST api and consumes json or yaml
- **Etcd** is the cluster store – an open source distributed key value database used to keep the state and config for the cluster
- We send a manifest file that defines our intents to the api server, the api server validates it, and saves it to the cluster store

Master in more detail

Develop Intelligence

- **kube-scheduler**
  - Component on the master that watches newly created pods that have no node assigned, and selects a node for them to run on
  - Factors taken into account for scheduling decisions include individual and collective resource requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference and deadlines
- **kube-controller-manager** - Component on the master that runs controllers
  - Node Controller: Responsible for noticing and responding when nodes go down
  - Replication Controller: Responsible for maintaining the correct number of pods for every replication controller object in the system
  - Endpoints Controller: Populates the Endpoints object (that is, joins Services & Pods)
  - Service Account & Token Controllers: Create default accounts and API access tokens for new namespaces

Control Plane Components: kube-apiserver

Develop Intelligence

- Exposes the API for the control plane
- It can scale as traffic increases

Control Plane Components: kube-controller-manager

Develop Intelligence

- Controllers are what keep the desired state maintained
- The kube-controller-manager runs all controllers in the cluster

Control Plane Components: cloud-controller-manager

Develop Intelligence

- Acts as liaison between the cluster and your cloud provider

Node Components: kubelet

Develop Intelligence

- Makes sure the pods in the node stay running and healthy
- They operate off the “spec” in a pod’s manifest YAML file

Node Components: kube-proxy

Develop Intelligence

- Manages network communication to pods
- Every pod has one as a network proxy

Node Components: Container Runtime

Develop Intelligence

- The software that runs the containers in the pod
- Docker is most common
- Others are containerd and CRI-O
- They need to comply with a standard called Kubernetes CRI

Addons: Cluster DNS

Develop Intelligence

- A DNS system that runs in a pod in your cluster
- It also spins up a k8s Service
- Allows the use of consistent DNS names instead of IP addresses

Addons: Dashboard

- Kubernetes-provided web-based UI dashboard for clusters

Lab: Set Up k8s Deployment on Local Machine

https://www.capitalone.com/tech/software-engineering/create-and-deploy-kubernetes-clusters/

Lab: Put your .NET Core App in a k8s Cluster

Develop Intelligence

- Use what you learned from to previous lab to make a cluster (3 nodes) that deploys your app from our Docker work

k8s Probes

Develop Intelligence

- The pods in a node are not always up and running, ready for incoming traffic
- We need a way to check
- The kubelet does this through periodic “liveness” checks into the pods in its node

kubectl

- kubectl (“kube-cuttle”) is command-line tool used to interact with API
- May require installation (automatic if using k8s via Docker Desktop)
- Config file defines details of connection to cluster (e.g., ~/.kube/config)
- Run `*kubectl cluster-info*` to confirm connectivity
- Run `*kubectl*` from command-line (with arguments) to see options
- NOTE: k8s in Docker Desktop should handle most of this for you

Pod Configuration

Develop Intelligence

- Two types of pod: single container and multiple container
- Making a single container pod is simple – use kubectl run

```
$ kubectl run <name of pod> --image=<name of the image from registry>
```

Pod Design Patterns

Develop Intelligence

- Usually, you will have one container per pod
- There are cases when multi-container pods make sense: When both pods have the same lifecycle, or they MUST run on the same node.
- An example: The primary container needs a helper process – it needs to be in the same node to be utilized.
- There are three main patterns for multi-container pods:
- Sidecar pattern
- Adapter pattern
- Ambassador pattern

Multi-container Pod: Sidecar Pattern

Develop Intelligence

- Main App plus a “helper” app that isn’t part of the main app
- Example: Main app is a web app; helper is a logging utility

Multi-container Pod: Adapter Pattern

Develop Intelligence

- Used when you need to standardize output form various apps in the cluster
- Each app type may format output in a unique way, but the cluster's monitoring needs to work with standardized formats
- Adding an adapter container to the pod for each app lets you still use the normal app output format in other areas, but have the cluster-level data work use a standardized format

Multi-container Pod: Ambassador Pattern

- One more way to allow the pod's containers to communicate with the outside world
- Add an "ambassador" container to the pod that acts as a proxy for network traffic to and from the primary container
- This lets you control traffic patterns based on the specific use case of the pod in question.
- Example: database connection in dev/qa/prod environments

Demo: Pods, Deployments, and Env Vars

Develop Intelligence

Lab: Use Environment Variables in Deployment

Diving Deeper into Kubernetes

Develop Intelligence

ConfigMaps vs. Environment Variables

Develop Intelligence

- A ConfigMap is an API object that lets you store configuration for other objects to use
- These are key/value pairs
- This lets you decouple environment-specific config data from your container images, making your apps more portable
- CAUTION: ConfigMaps do not provide encryption or secrecy. Use a Kubernetes Secret for that.

ConfigMaps

Develop Intelligence

- Can be:
  - Set of key-value pairs
  - Blurb of text
  - Binary files
- One pod can use many ConfigMaps and one ConfigMap can be used by many pods
- Data is read-only – pod can’t alter

ConfigMaps as Env Vars

Develop Intelligence

- Can be created from a literal:

`*kubectl create configmap <name> --from-literal=<key>=<value>*`

- Can be created from a file (to group together multiple settings):

`*kubectl create configmap <name> --from-env-file=<file-path>*`

Where <file-path> contains .env file like:

key1=value1

key2=value2

- Can be created from a .yml definition file (like all things k8s)

ConfigMaps

Develop Intelligence

- Can be presented as files inside directories in the container
- Uses volumes – making contents of ConfigMap available to pod
- Uses volume mounts – loads contents of ConfigMap volume into specific container path in pod
- Can contain settings to override defaults

ConfigMaps - Precedence

Develop Intelligence

- If env vars defined in multiple places, definitions in `env` section in pod spec will trump others (localized)
- Typical approach:
  - Default app settings “baked in” to container image (e.g., to support dev mode)
  - Specific settings for an environment stored in ConfigMap and surfaced to container filesystem
  - Merges with default settings (overwriting where applicable)
  - Final tweaks can be accomplished with env variables in pod spec

Demo: ConfigMaps

Develop Intelligence

Lab: ConfigMaps

Develop Intelligence

Secrets in Pods

Develop Intelligence

- Secrets let you store and manage sensitive information, such as passwords, OAuth tokens, and ssh keys.
- A pod needs to reference a secret. There are three ways get to the secret:
- As a file in a Volume that is mounted to a container;
- As an environment variable for a container;
- By the kubelet, when it pulls images for the pod
- NOTE: The default config for a Secret does NOT have encryption – data is plain text. You need to configure "Encryption at Rest" for the Secret", and Role Based Access Control in your cluster

Demo: Set Up Secrets for Deployment

Develop Intelligence

Lab: Set Up Secrets for Deployment

Develop Intelligence

k8s Services

Develop Intelligence

- Services are an abstraction
- They define a set of pods, and an access policy for them
- A pod has an IP address
- Pods are ephemeral – what happens to IP traffic if they die?
- Pods in a service can have fixed IP addresses that stay the same even if the actual pod dies and is spun up again
- Services are k8s objects; they are created like any other

k8s Service Types

Develop Intelligence

- ClusterIP (default): IP is internal to the cluster
- NodePort: IP for the port is exposed with a static IP address
- LoadBalancer: IP traffic to nodes is managed by external load balancer
- ExternalName: IP traffic is routed by DNS Name (foo.example.com)

Cluster Access – Internal vs. External

Develop Intelligence

- You can access the cluster API using kubectl (“proxy”)
- You can get libraries for popular languages that can access the cluster
- k8s objects (pods) can also access the API internally
- This uses an internal DNS
- “kubernetes.default.svc” maps to the API server
- The pod usually uses a “sidecar” container to do this

Demo: Using Services in k8s

Develop Intelligence

Lab: Using Services in k8s

Develop Intelligence

Container File System

- Each container in a pod has its own filesystem built by k8s
- Can be built from multiple sources, including:
  - Layers from the container image
  - Writable layer for the container
- Can be expanded with other sources like ConfigMaps and Secrets – mounted to a specific path in a container
- Volumes can provide another source of storage – volumes are defined and mounted to containers

EmptyDir

- Empty directory stored at pod level vs. container level
- Mounted as a volume into the container so visible there
- Any data stored there from the container remains in pod between restarts
- Replacement containers can access data from predecessors

HostPath

Develop Intelligence

- Data physically stored on node in cluster
- Extends beyond lifecycle of pod
- Available to replacement pods (but only if run on same node)
- Mounted as a volume into the container like others
- However, can have issues:
  - In cluster with multiple nodes, limits usefulness
  - If not careful, can expose large blocks of host data

PersistentVolume

Develop Intelligence

- Like pods (abstraction over computer) and services (abstraction over network), persistent volumes provide abstraction over storage
- k8s object that defines available section of storage
- Can be “mapped” to shared storage (like NFS) or locally (for dev/test purposes)

PersistentVolumeClaim

Develop Intelligence

- Pods are not allowed to use persistent volumes directly
- Instead, pods claim using PersistentVolumeClaim object type in k8s
- Requests storage for a pod
- k8s handles matching up a claim to a persistent volume
- Provides virtual storage abstracted from actual storage

Dynamic Provisioning

Develop Intelligence

- Static provisioning accomplished by explicitly creating persistent volume and then claim (k8s binds claim to volume)
- Dynamic provisioning supported as well
- You create the persistent volume claim and let persistent volume be created on demand by cluster
- Can leverage storage classes for defining and matching types of storage

Storage Classes

- Defined by three properties:
  - provisioner – component that creates persistent volumes on demand
  - reclaimPolicy – what to do with dynamically created volumes when claim is deleted
  - volumeBindingMode – eager vs. lazy binding/creation (at same time as claim creation or only when pod using the claim get created)

Storage Classes

Develop Intelligence

- Able to create custom storage classes
- Defined by three properties:
  - provisioner – component that creates persistent volumes on demand
  - reclaimPolicy – what to do with dynamically created volumes when claim is deleted
  - volumeBindingMode – eager vs. lazy binding/creation (at same time as claim creation or only when pod using the claim get created)
- Remember – persistent data can limit statelessness; be aware and use wisely

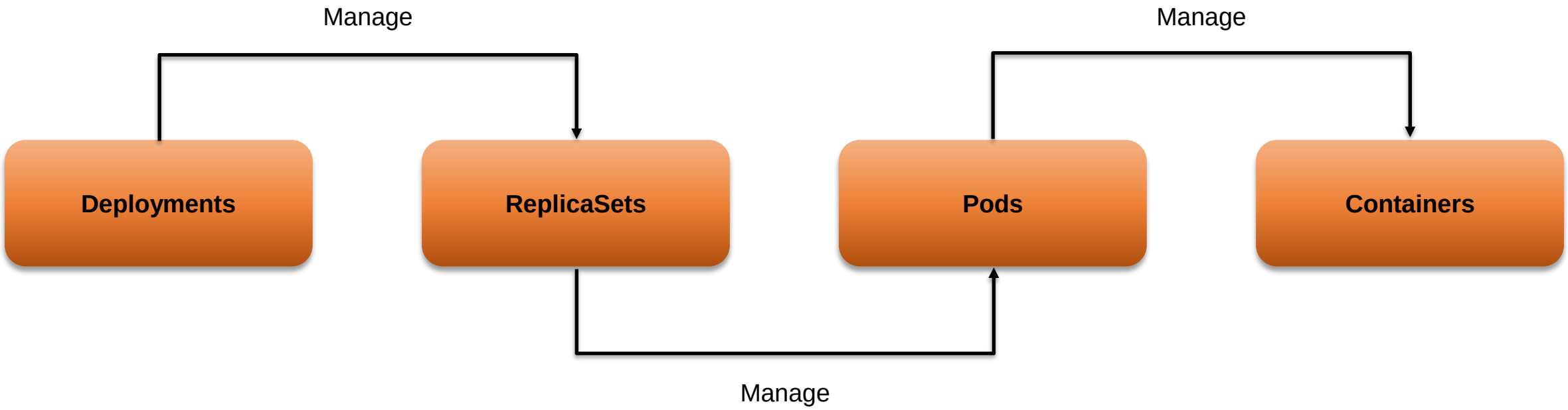
Demo: Using Volumes in k8s

Develop Intelligence

Lab: Using Volumes in k8s

Develop Intelligence

Scaling k8s


Manage
Manage
Deployments
ReplicaSets
Pods
Containers
Manage

ReplicaSets

Develop Intelligence

- Can be created directly as an object in k8s
- However, better practice would be to create/manage through deployments
- `replicas` property in pod spec in deployment can be used to build out multiple replicas for a pod config
- If not included, defaults to 1

Setting Limits in Deployments

- When defining pod spec in deployment, can use limits to manage
- `resources` attribute defines compute resource requirements for pod
- `limits` can be used to apply limits (e.g., in amount of memory and cpu)
- Usually better to optimize once running vs. prematurely optimizing
- Target environment when running will help determine correct settings

Deploying and Scaling Clusters

- Clusters can scale as needed
  - Increased load
  - Rolling cluster updates
- This is often managed by the cloud provider with cloud native

The Ingress Controller

- Specialized load balancer for k8s environments
- Manages traffic to pods in your cluster
- Like Nodes, Services etc., they are deployed using the cluster API
- There are several available besides the k8s offering

Demo: k8s Deployment in AWS

Develop Intelligence

https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html

Lab: k8s Deployment in AWS

Develop Intelligence

Work through this deployment of a cluster on AWS:
https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html

A sandbox with a username/password has been created for you

Rolling Deployments

- Code changes, so your pod (containers) change
- We want a stable experience for users
- k8s does this via rolling deployments
- As updates nodes become available, traffic is load balanced to ensure uptime

Blue/Green Deployments

Develop Intelligence

- How they work: keep two identical environments, conventionally called blue and green, to do continuous, risk-free updates. This way, users access one while the other receives updates.
- With bare metal servers, this is expensive – one of the environments isn't used much when the other is
- k8s makes this easy – we can make a second environment, switch traffic over, then kill the first with little impact

Canary Pods for Testing and Production

Develop Intelligence

- A strategy to try out a new feature only for a small segment of users or traffic (A/B testing)
- k8s makes this easy
- Use a tag to designate what pods will receive a small percentage of traffic
- Also, allows smoke testing on small scale before “opening floodgates”

Review

Develop Intelligence

Docker Fundamentals

Develop Intelligence

- What is Docker?
- What does it enable in development? Testing? Deployment?
- How does it work under the hood?

- How can we use containers in CI/CD pipelines?
- How can we make smaller, more efficient images?
- How can we make multi-stage builds – and why should we?
- What is the OCI?
- Why should we have alternatives to Docker runtime?

Kubernetes Fundamentals

- Why would we need container orchestration?
- How does k8s work?
- What to we use environment variables for?
- How do we handle sensitive info in k8s?

Advanced Kubernetes

Develop Intelligence

- What are k8s Services? What do we use them for?
- How do we scale Pods?
- What is the Ingress Controller? Why do we have alternatives to it?
- What are some deployment strategies for k8s? What should we think about when choosing one?
- What are “canary pods” and when can we use them?

Future Study

Develop Intelligence

THANK YOU

Develop
Intelligence