Docker from Dev to Prod

for JavaSkop 2015

Chris

@cimnine

Disclaimer: The opinions expressed are those of the speaker and do not reflect opinions of any other person, company or agency.

Chris

@cimnine

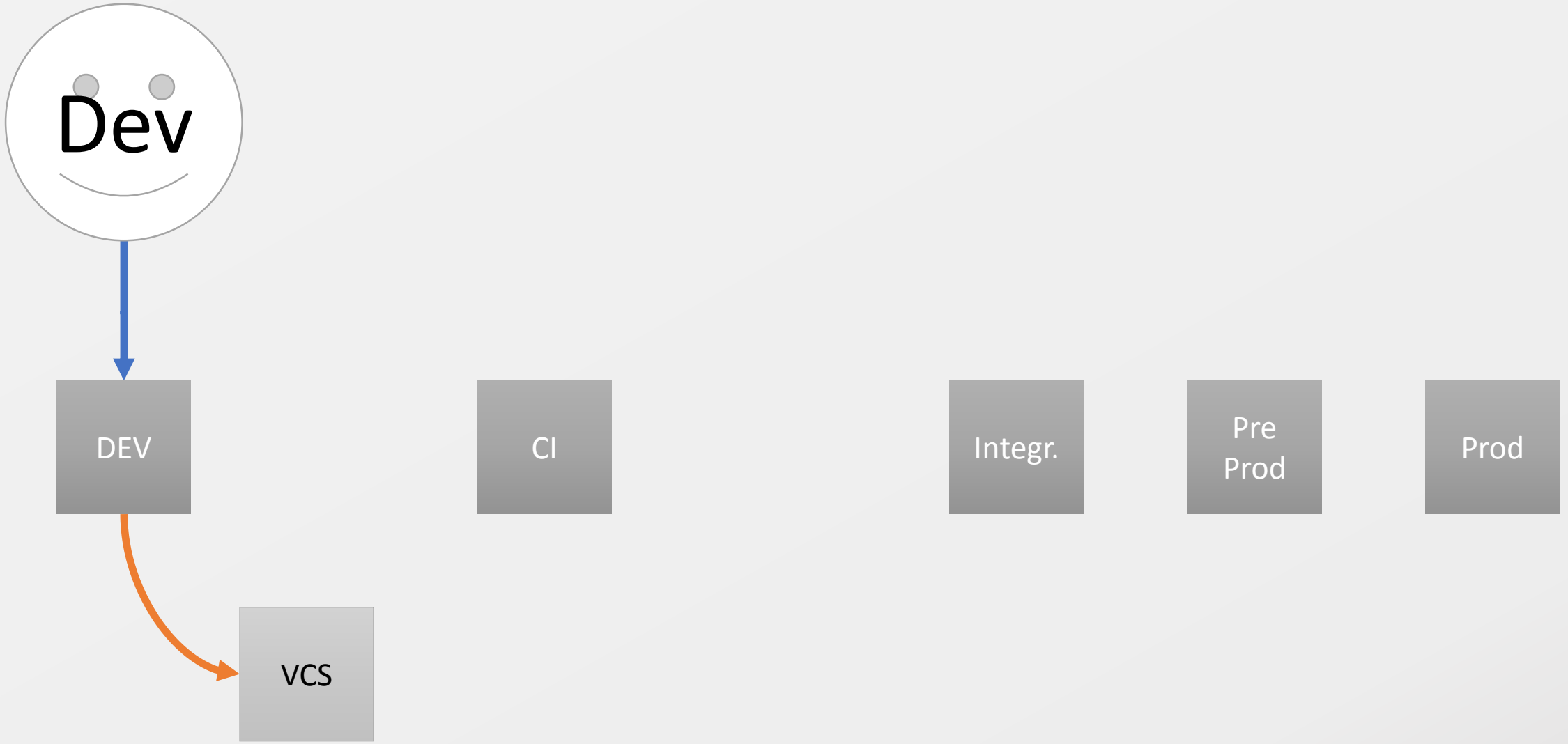
Disclaimer: The opinions expressed are those of the speaker and do not reflect opinions of any other person, company or agency.

Chris

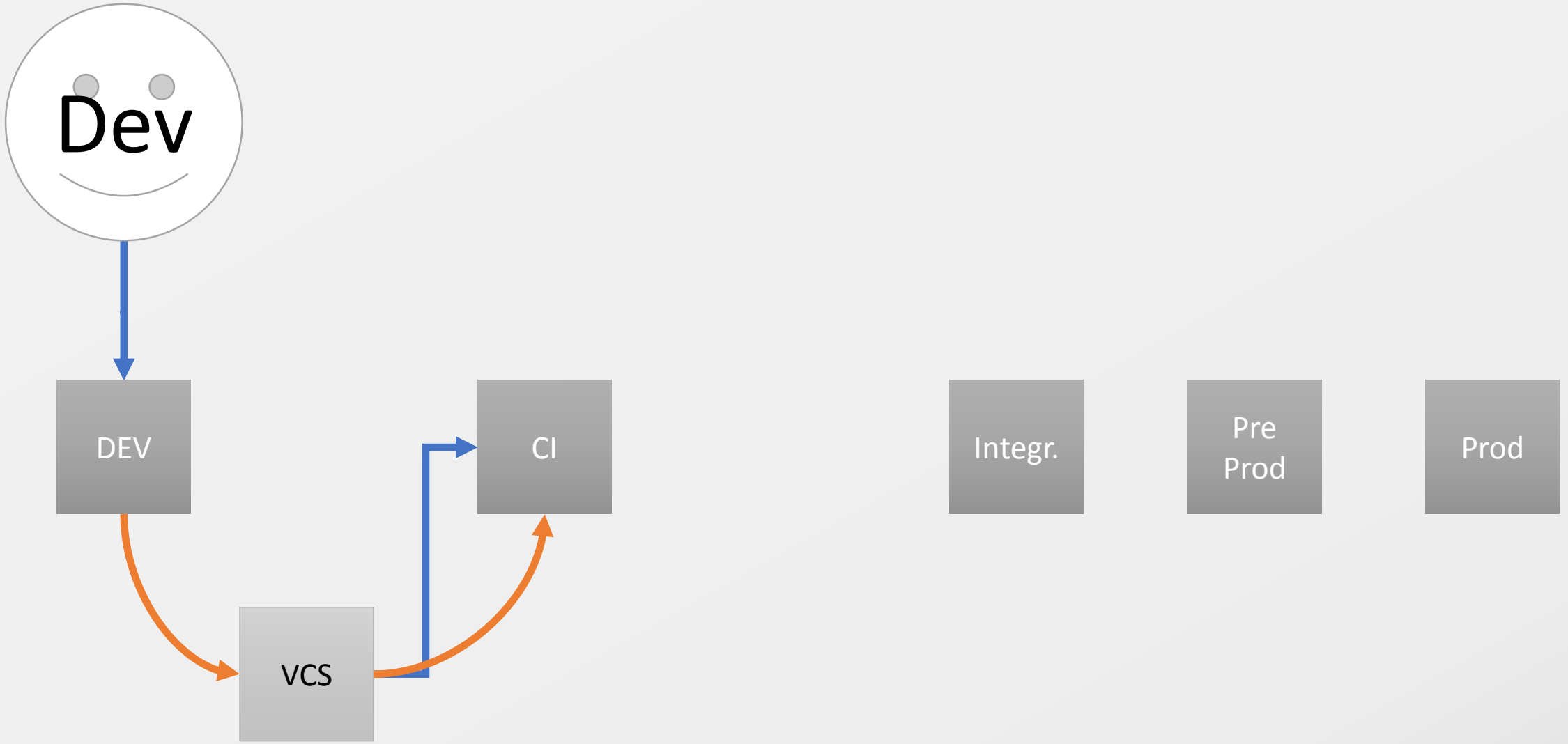
@cimnine

Disclaimer: The opinions expressed are those of the speaker and do not reflect opinions of any other person, company or agency.

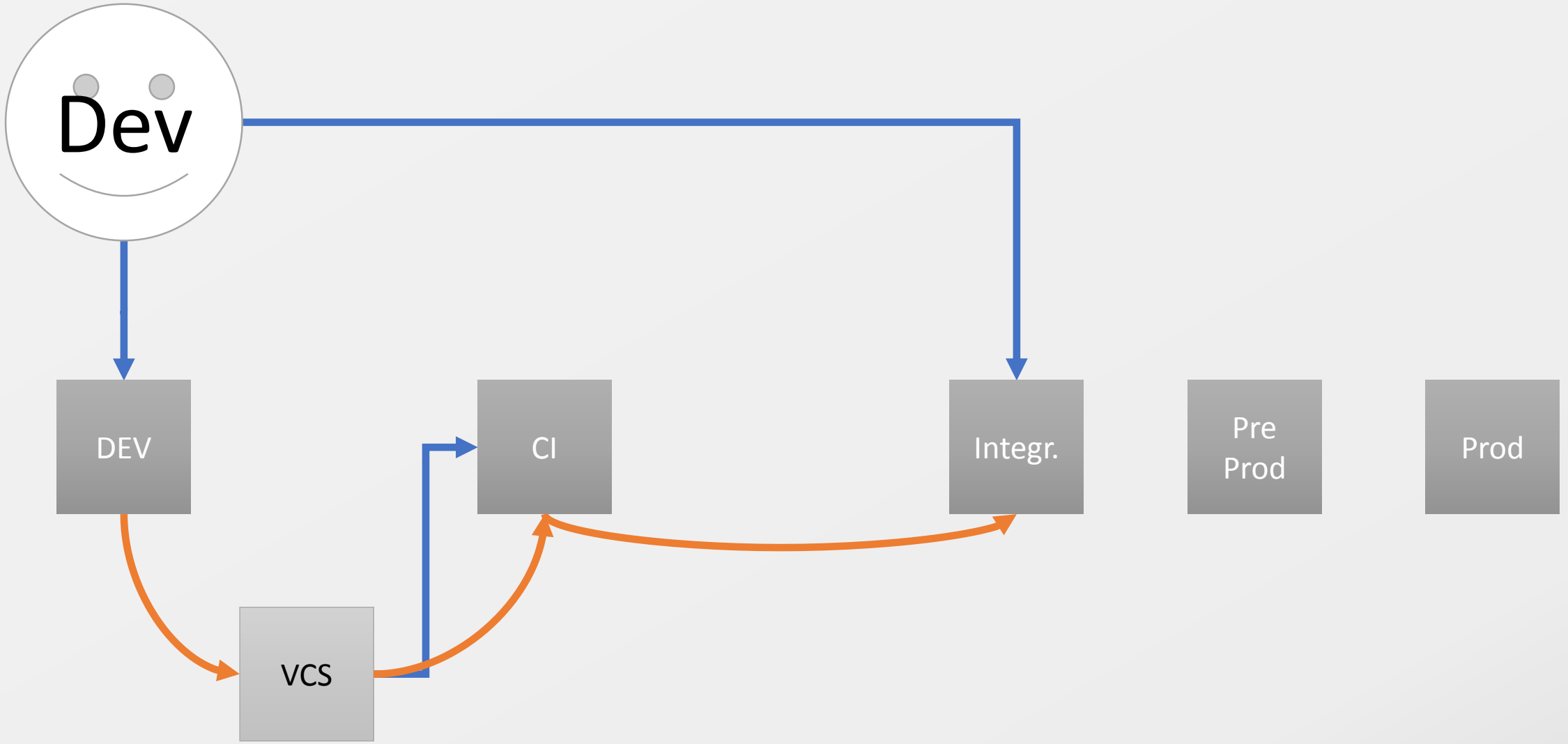

Dev
CI
Integration
Pre-Prod
Prod



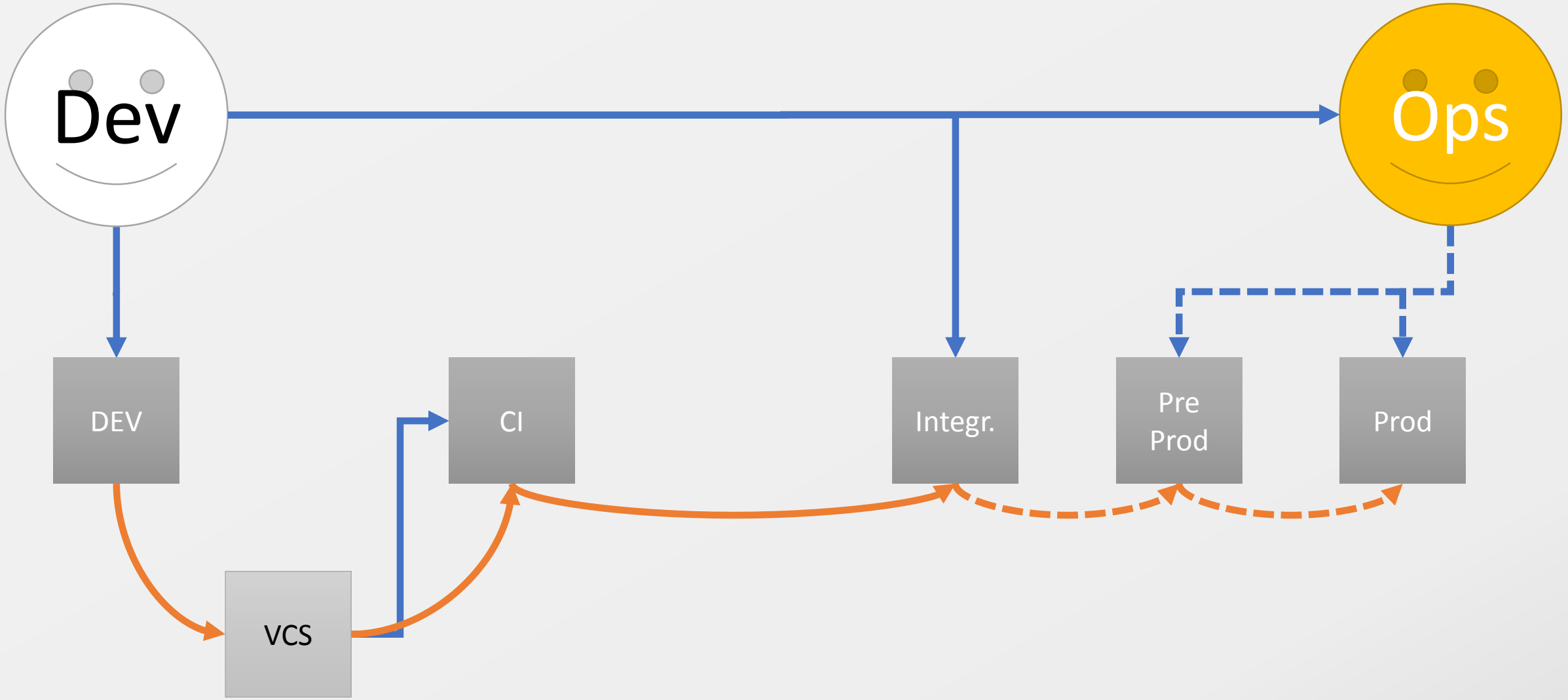
orange lines are data flow and **blue lines are control flow**



orange lines are data flow and **blue lines are control flow**

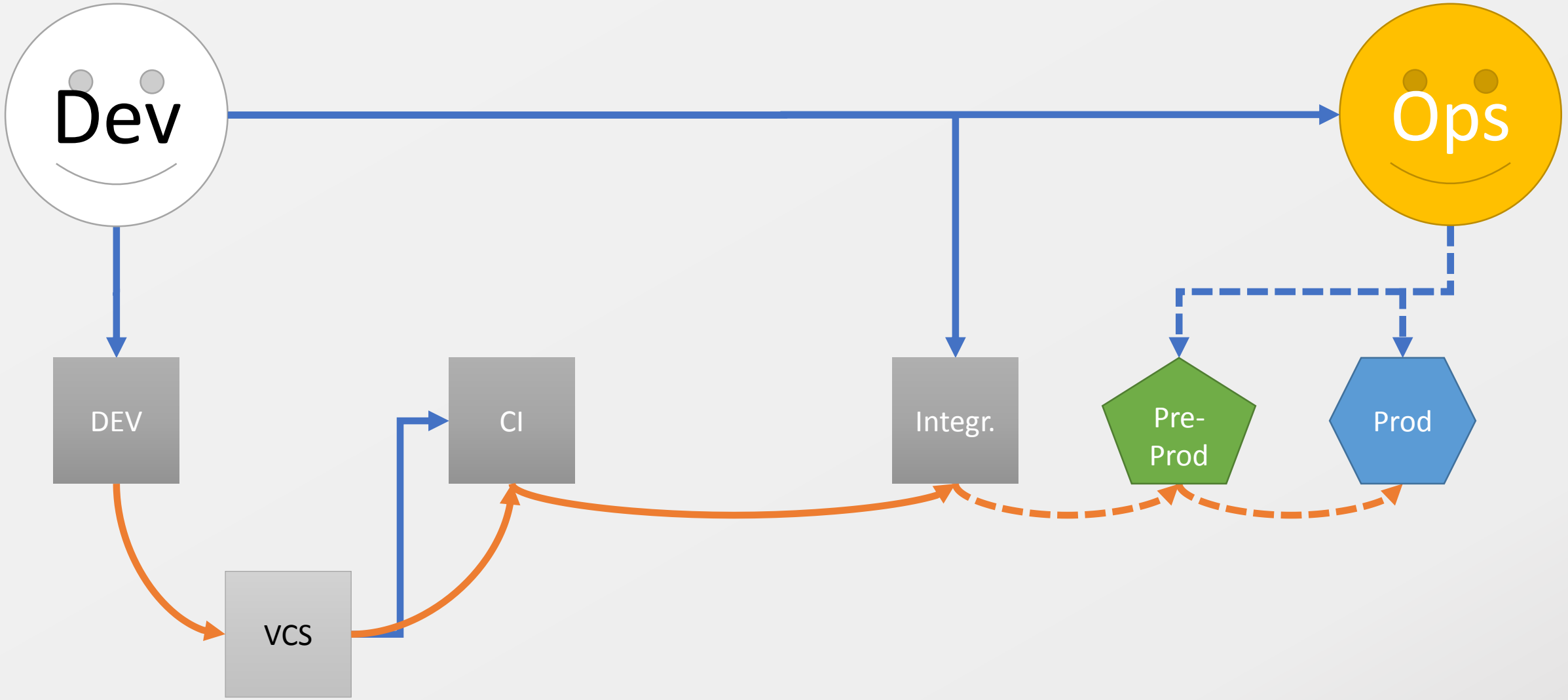


orange lines are data flow and **blue lines are control flow**



orange lines are data flow and **blue lines are control flow**


WORKED FINE IN DEV
OPS PROBLEM NOW
memegenerator.net

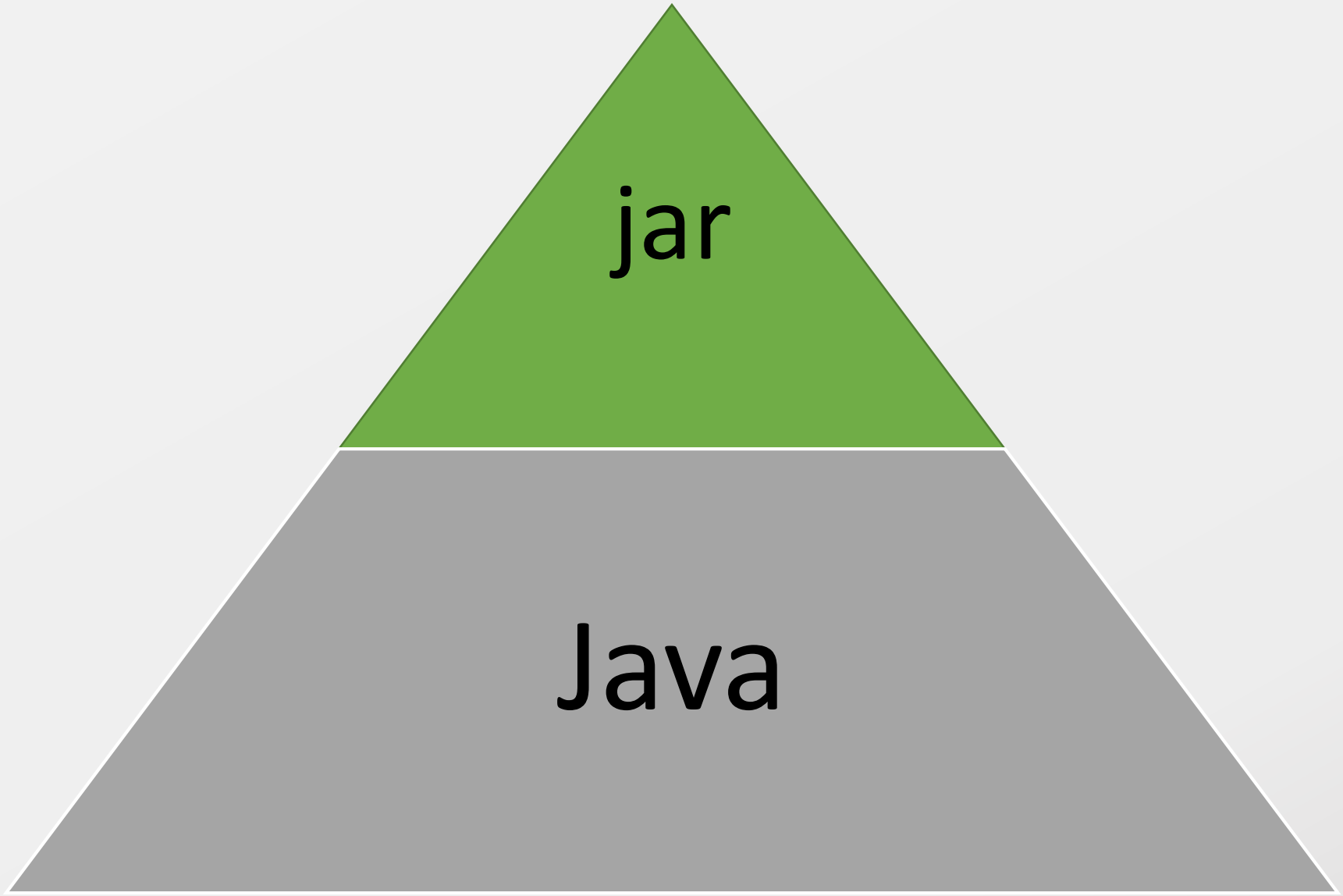


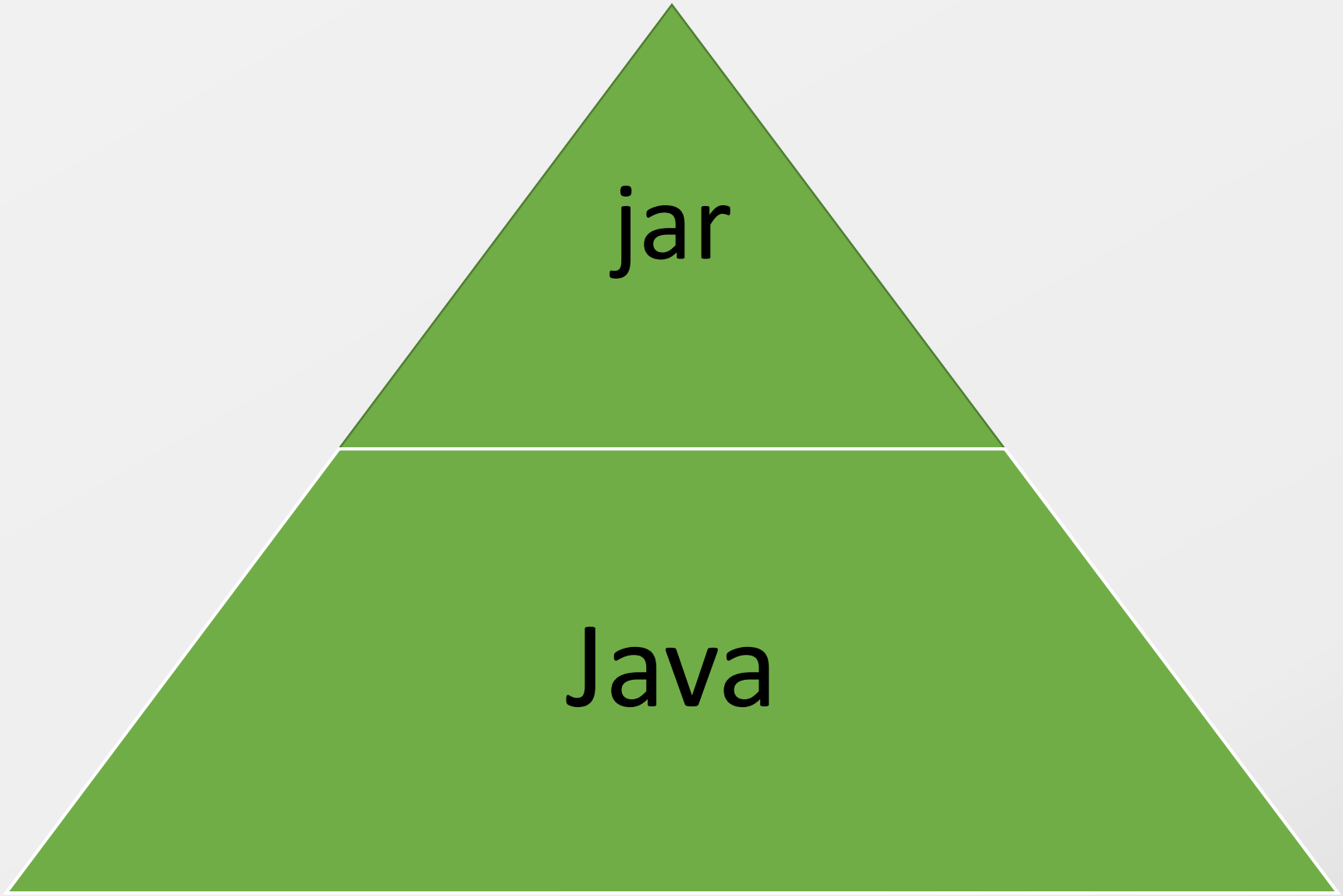
orange lines are data flow and **blue lines are control flow**

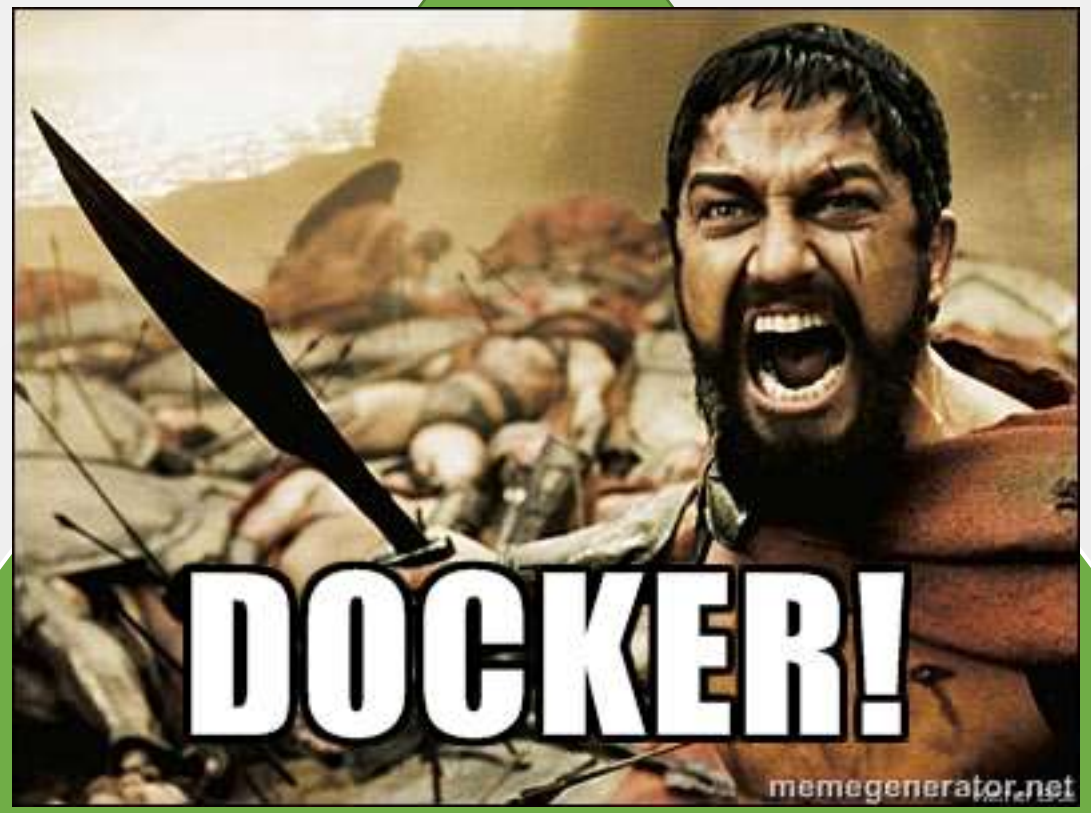

Dev
CI
Integration
Pre-Prod
Prod


Dev
CI
Integration
Pre-Prod
Prod

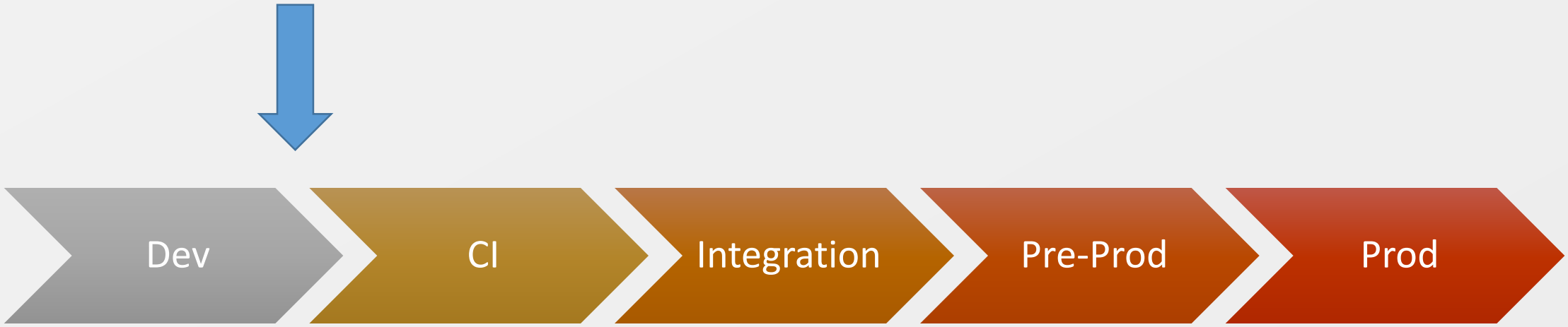

Dev
CI
Integration
Pre-Prod
Prod


jar
Java


jar
Java


DOCKER!
memegenerator.net


Dev
CI
Integration
Pre-Prod
Prod

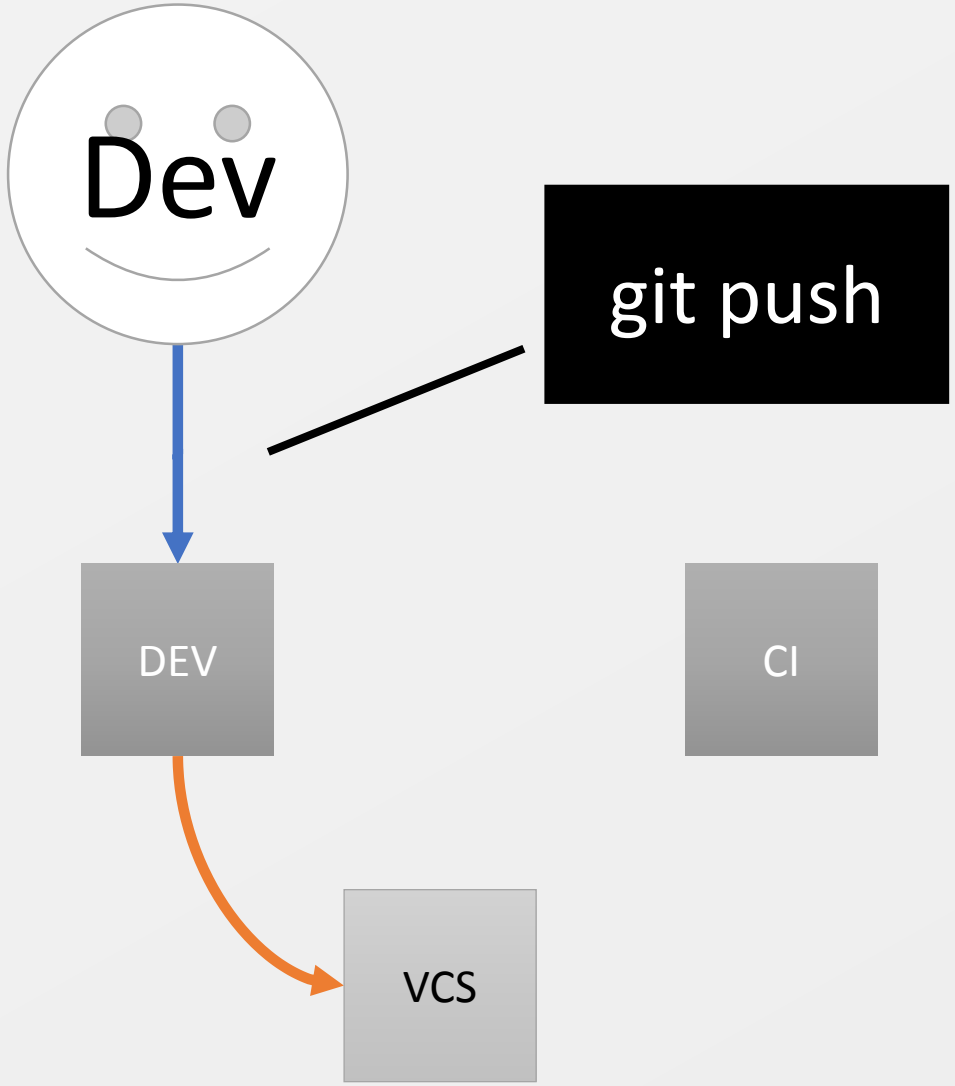

Dev
CI
Integration
Pre-Prod
Prod

DEV

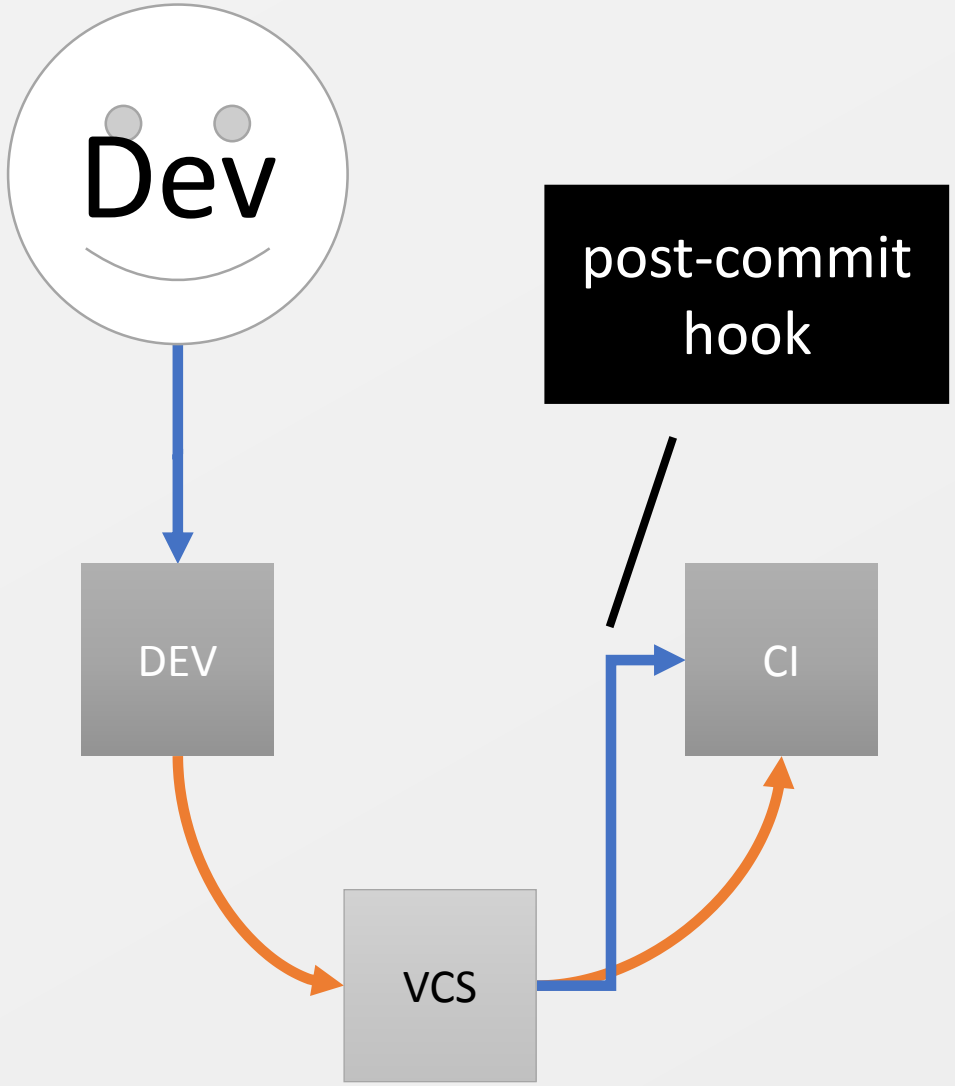
CI

VCS

orange lines are data flow and blue lines are control flow



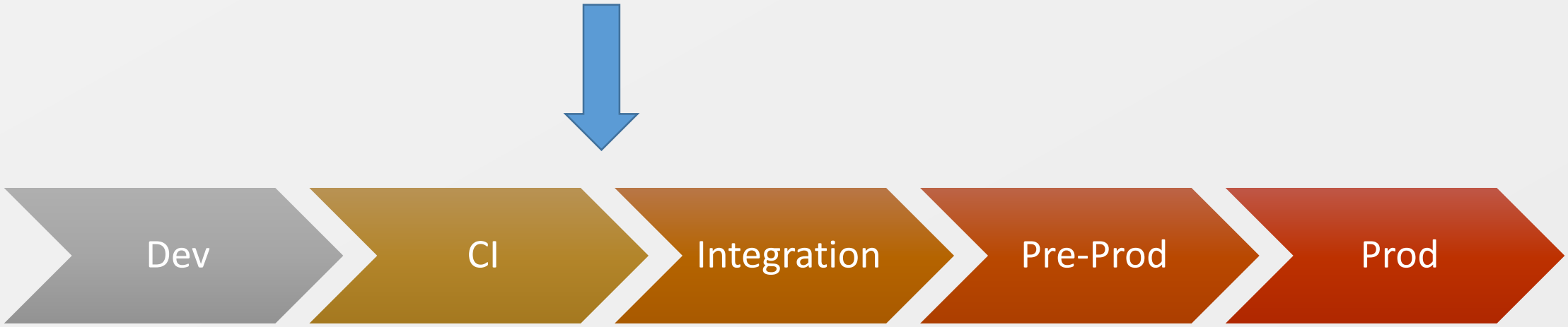
orange lines are data flow and **blue lines are control flow**

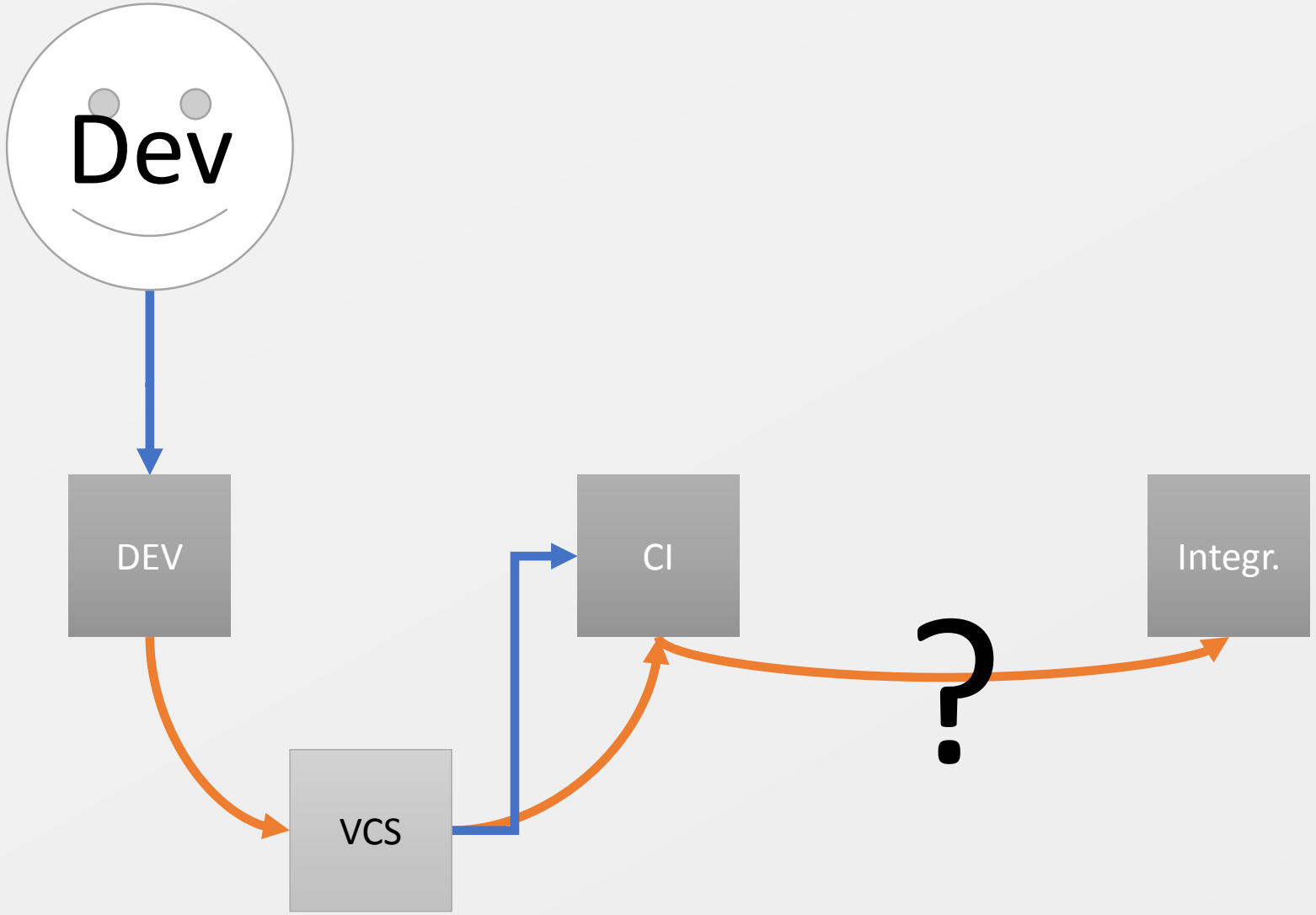


orange lines are data flow and blue lines are control flow

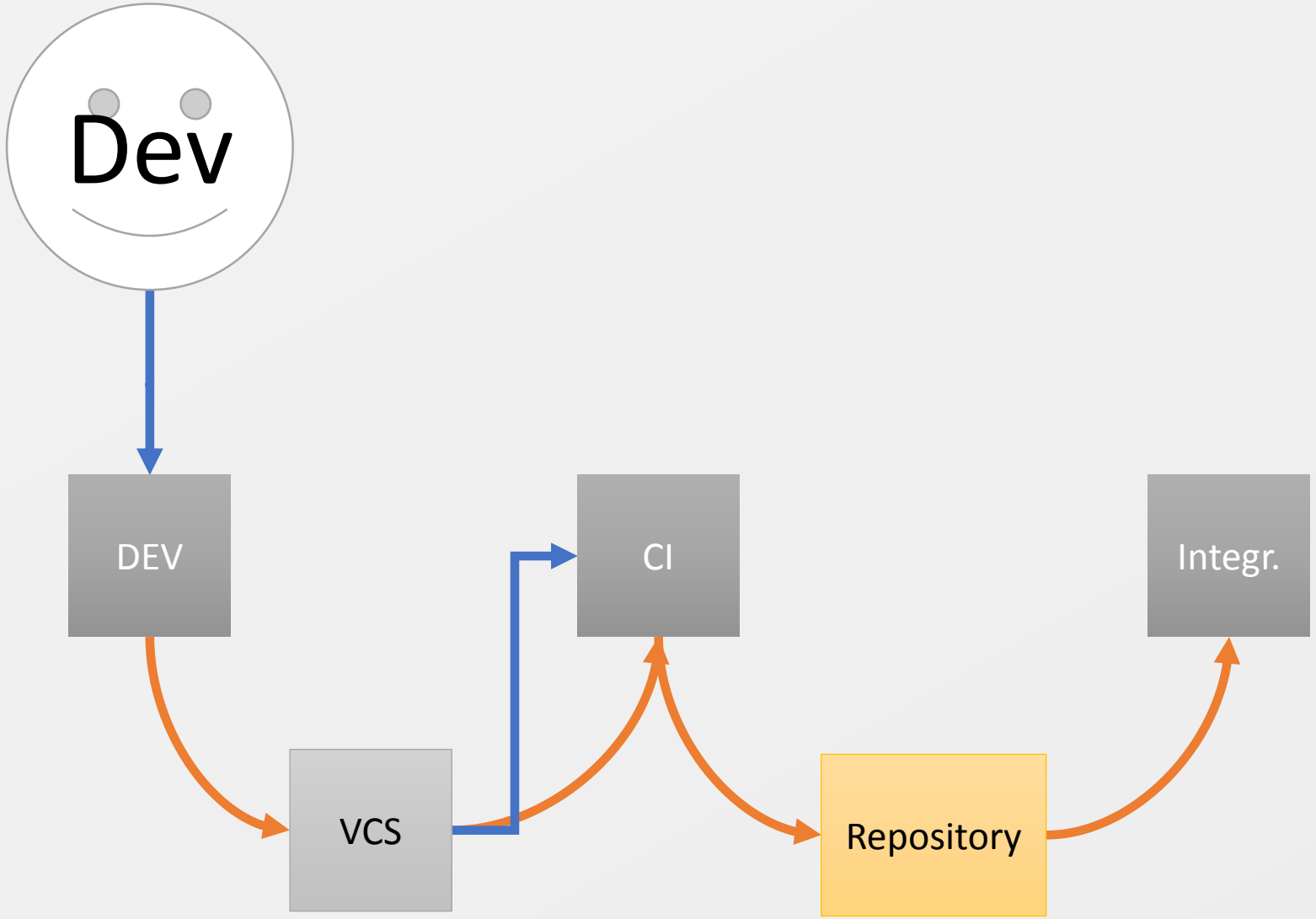

Dev
CI
Integration
Pre-Prod
Prod


Dev
CI
Integration
Pre-Prod
Prod

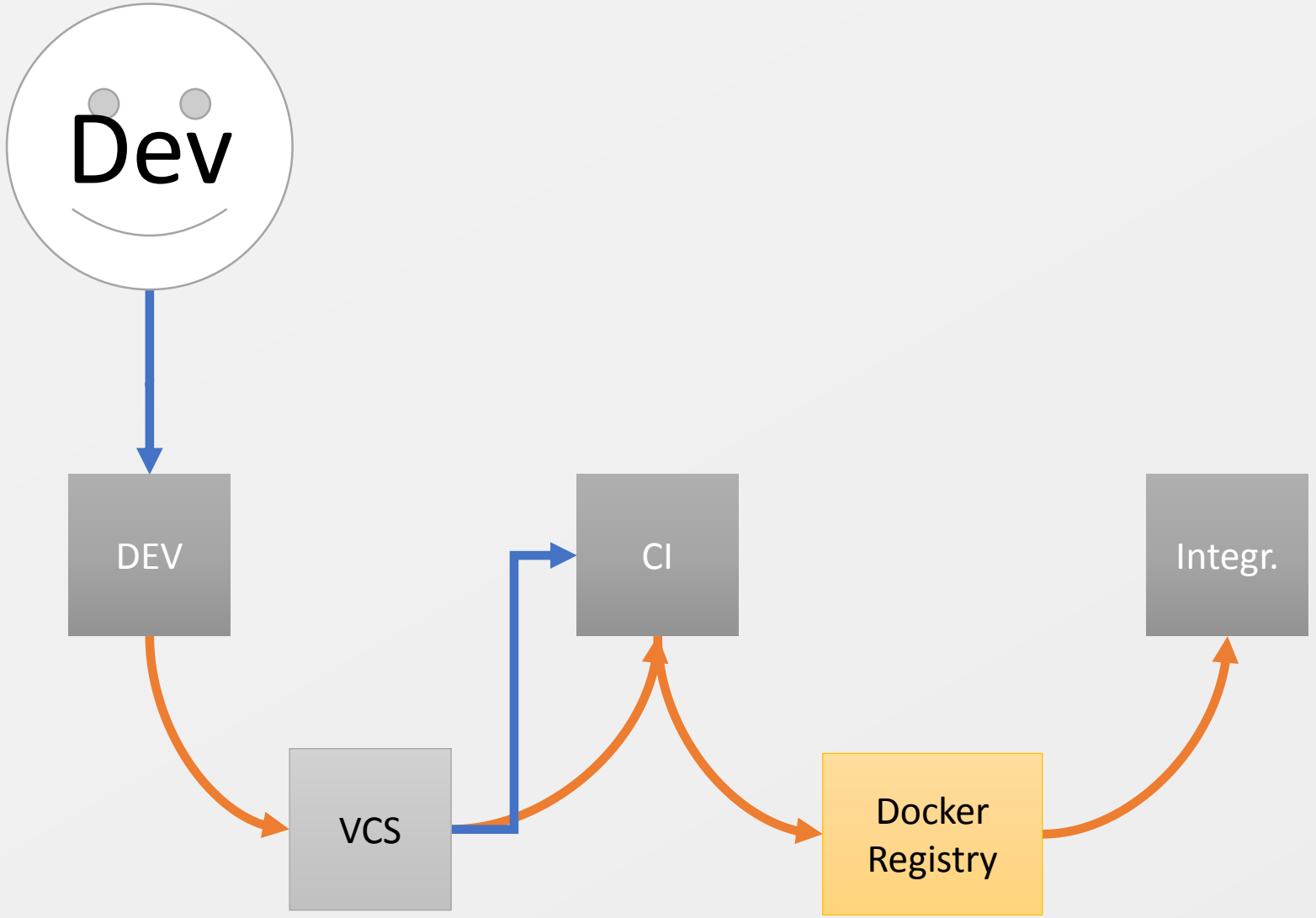

Dev
CI
Integration
Pre-Prod
Prod



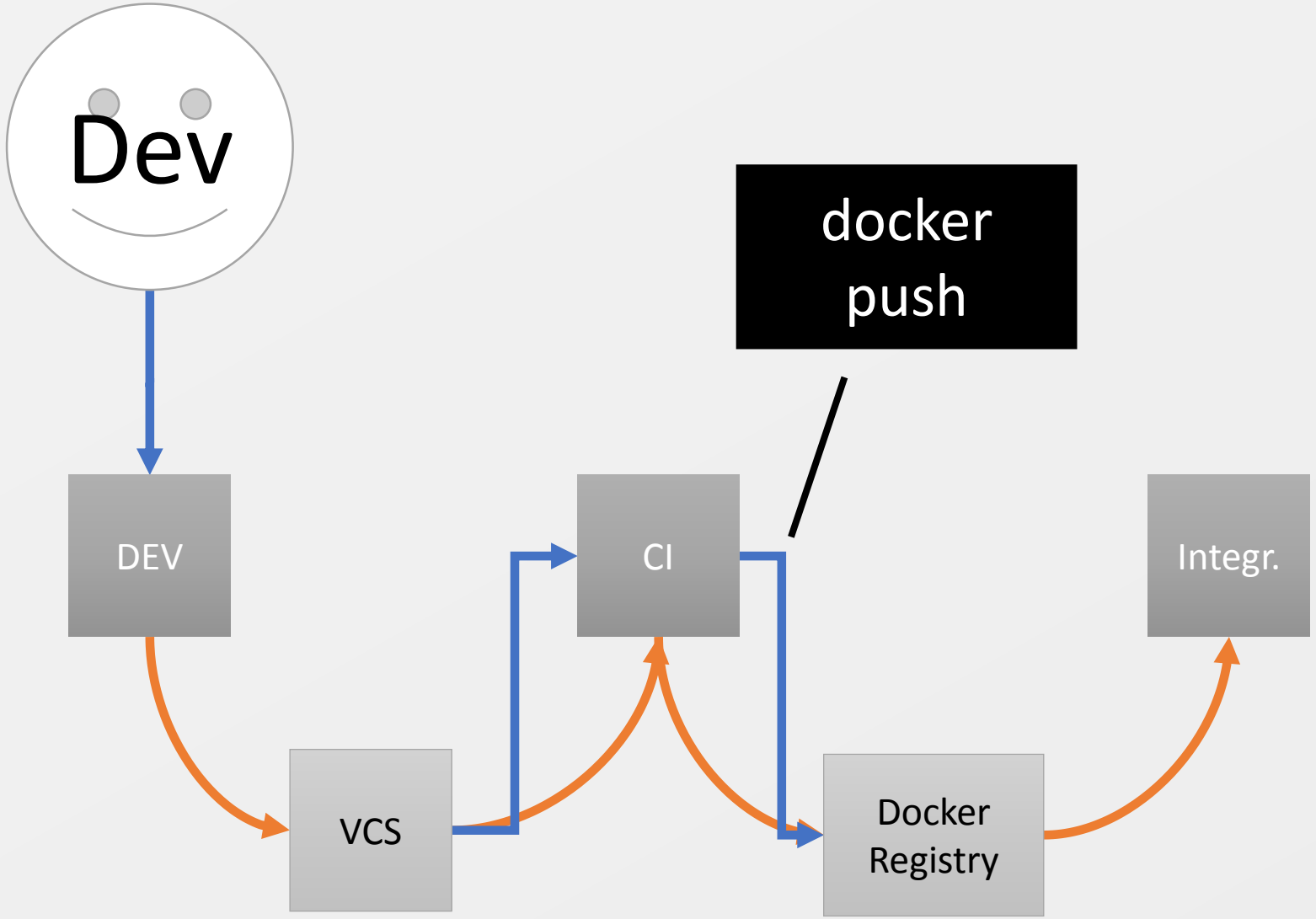
orange lines are data flow and **blue lines are control flow**



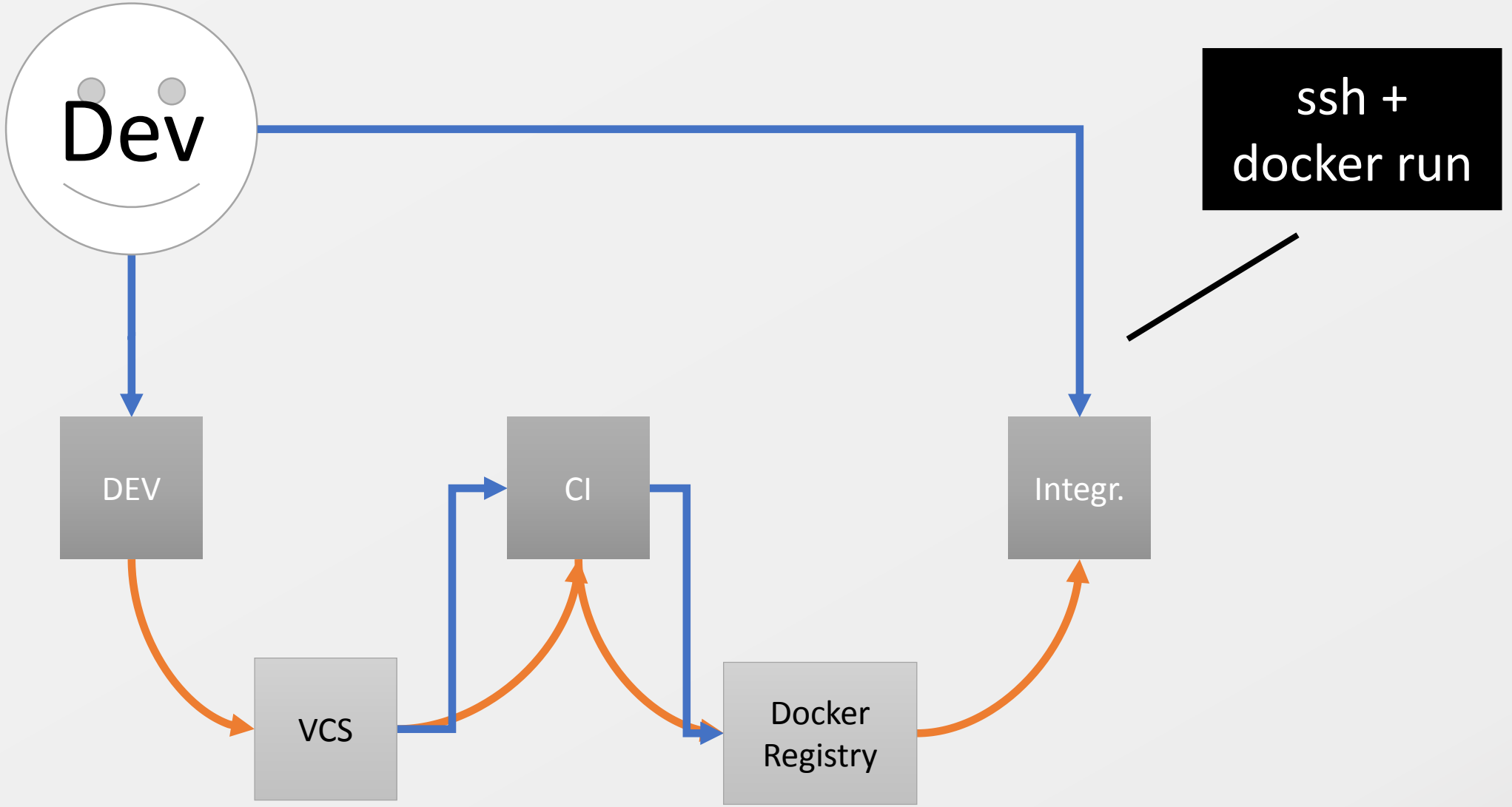
orange lines are data flow and **blue lines are control flow**



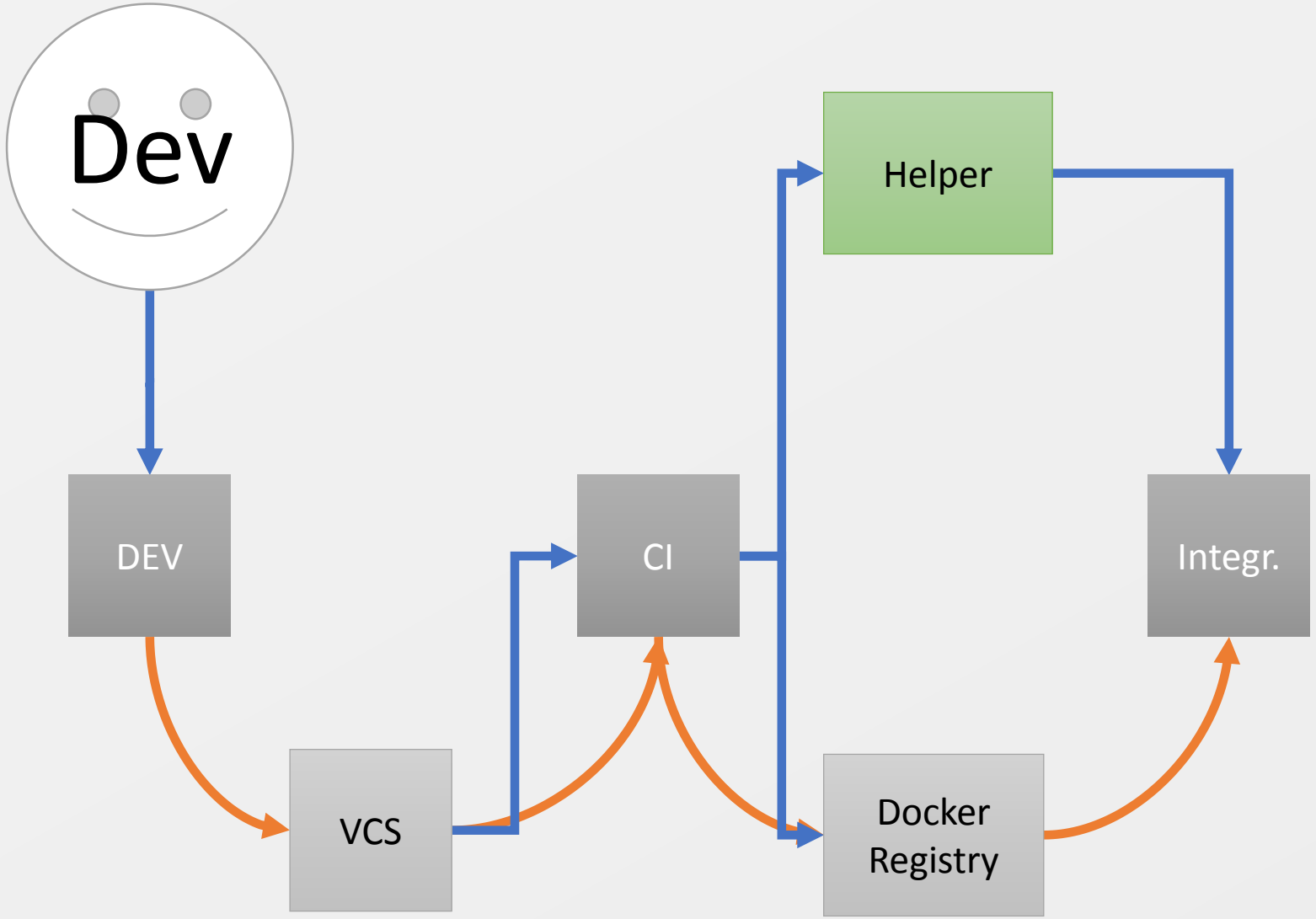
orange lines are data flow and **blue lines are control flow**



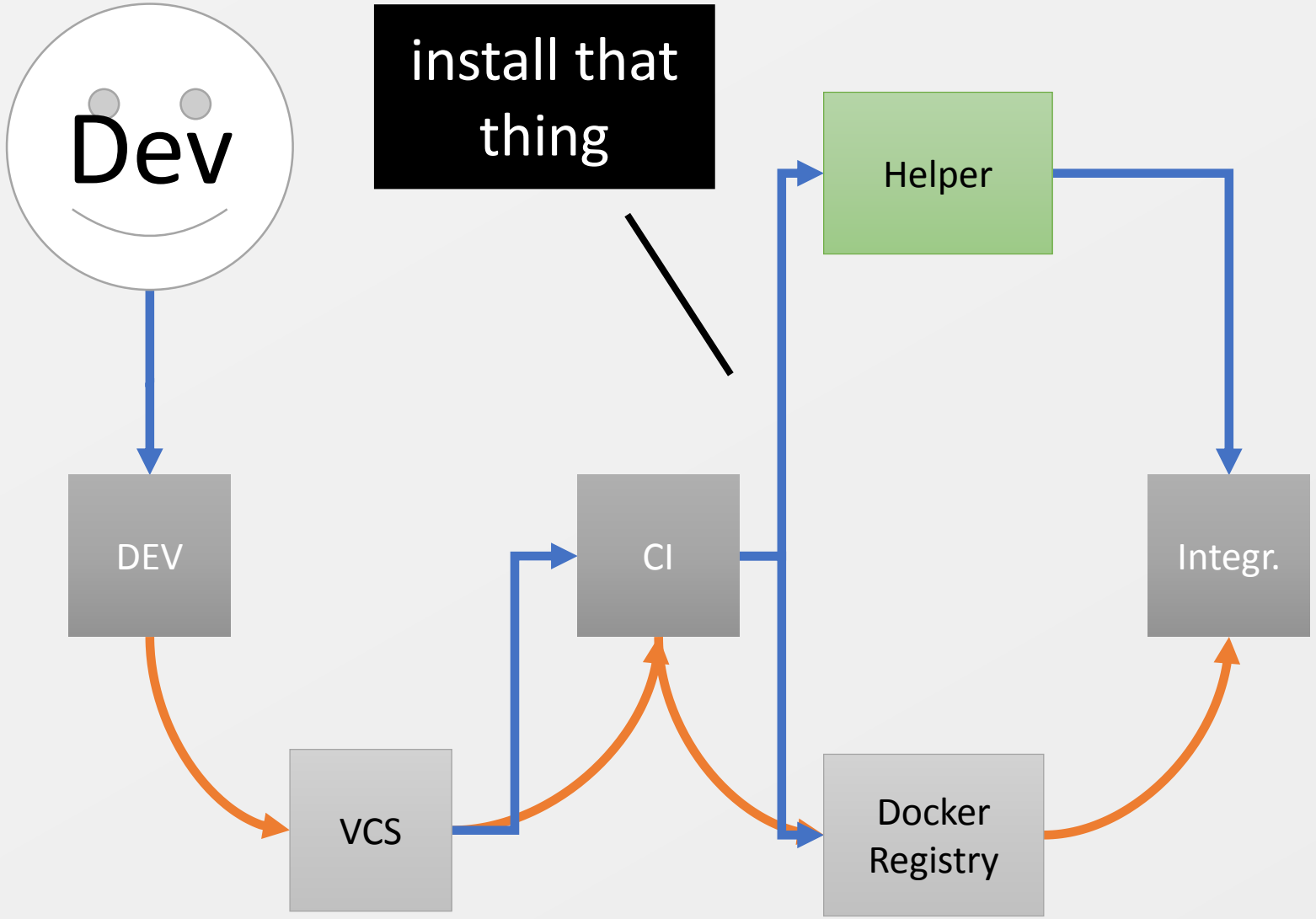
orange lines are data flow and **blue lines are control flow**



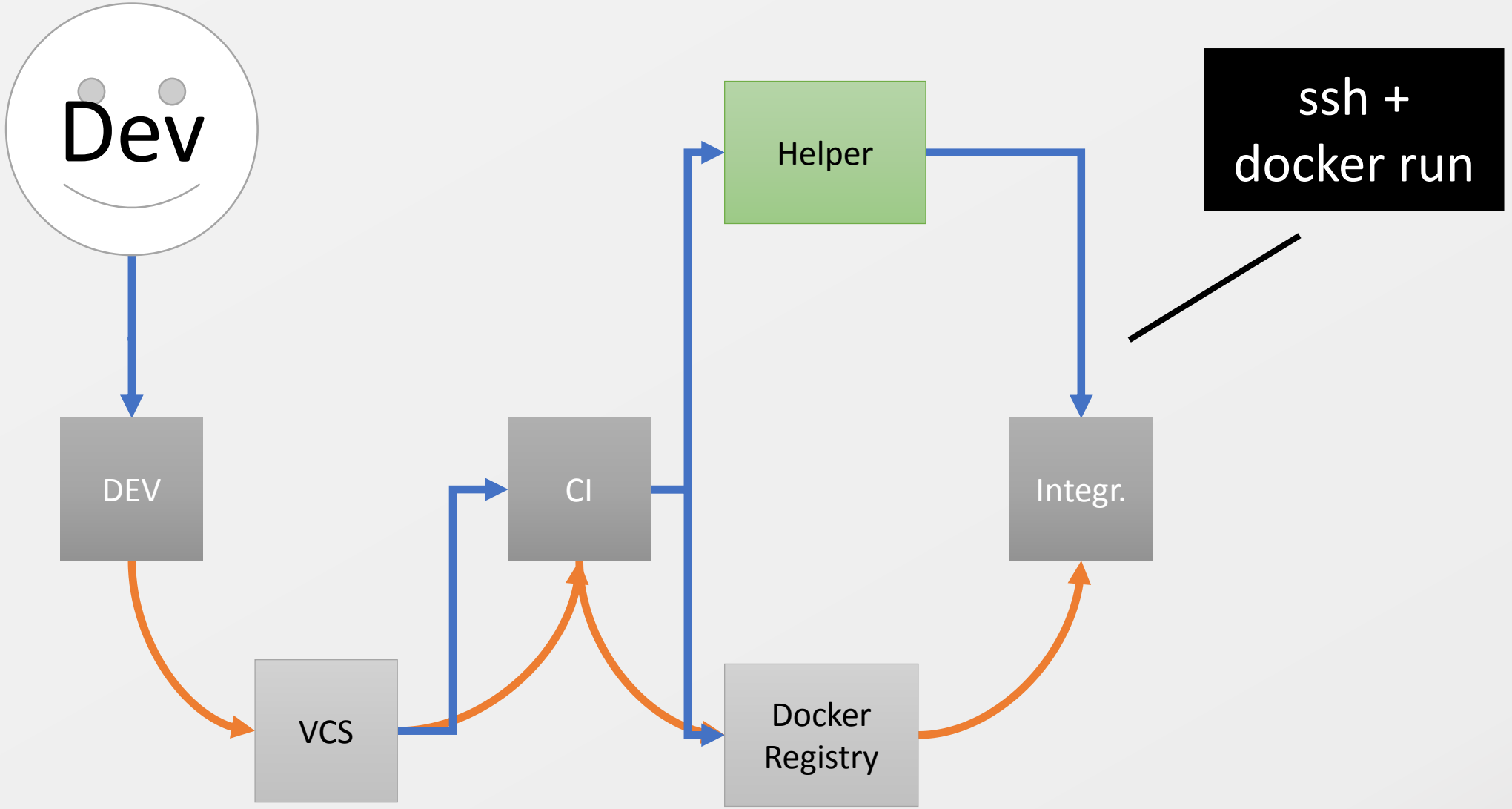
orange lines are data flow and **blue lines are control flow**



orange lines are data flow and **blue lines are control flow**



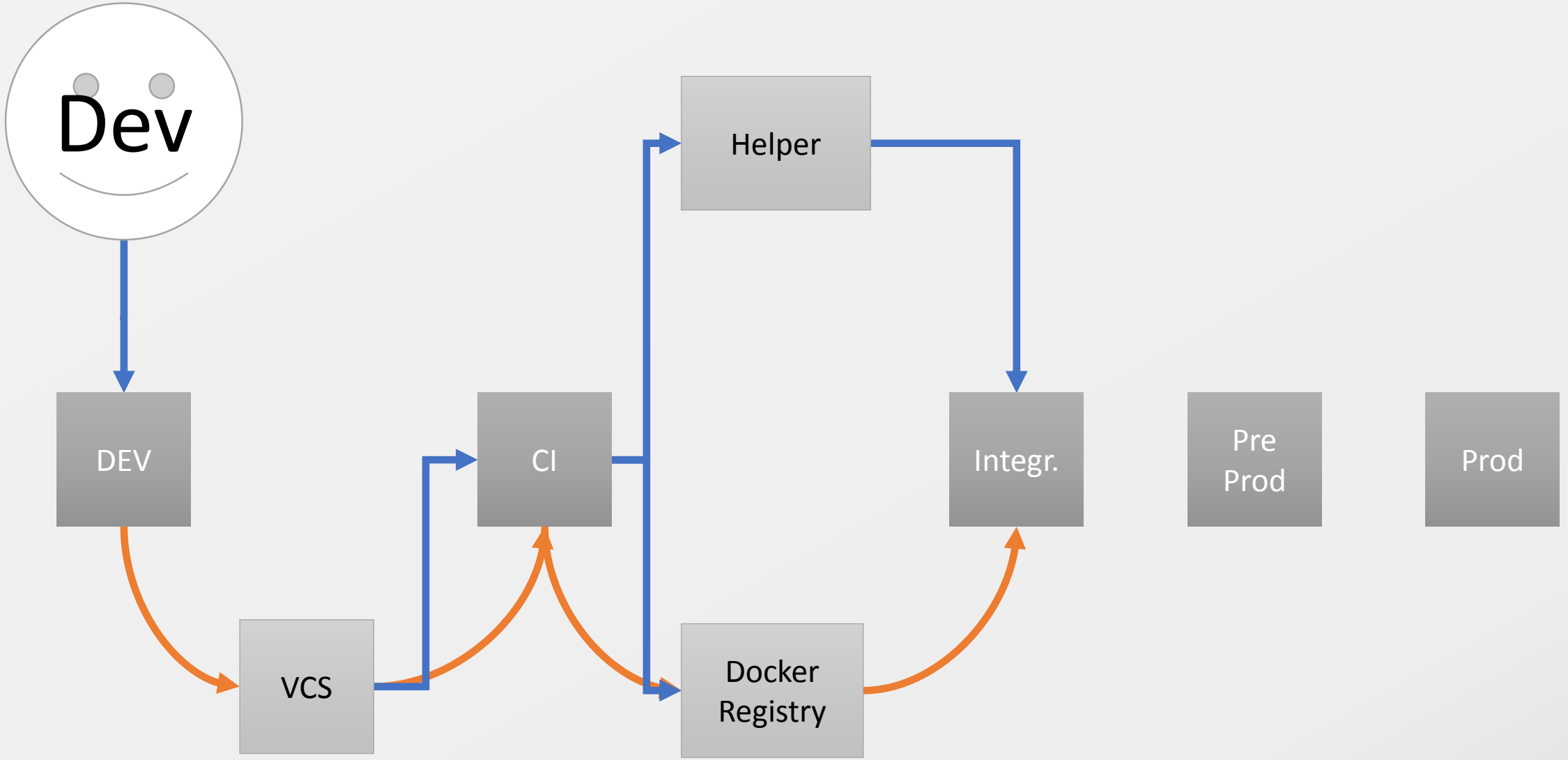
orange lines are data flow and **blue lines are control flow**



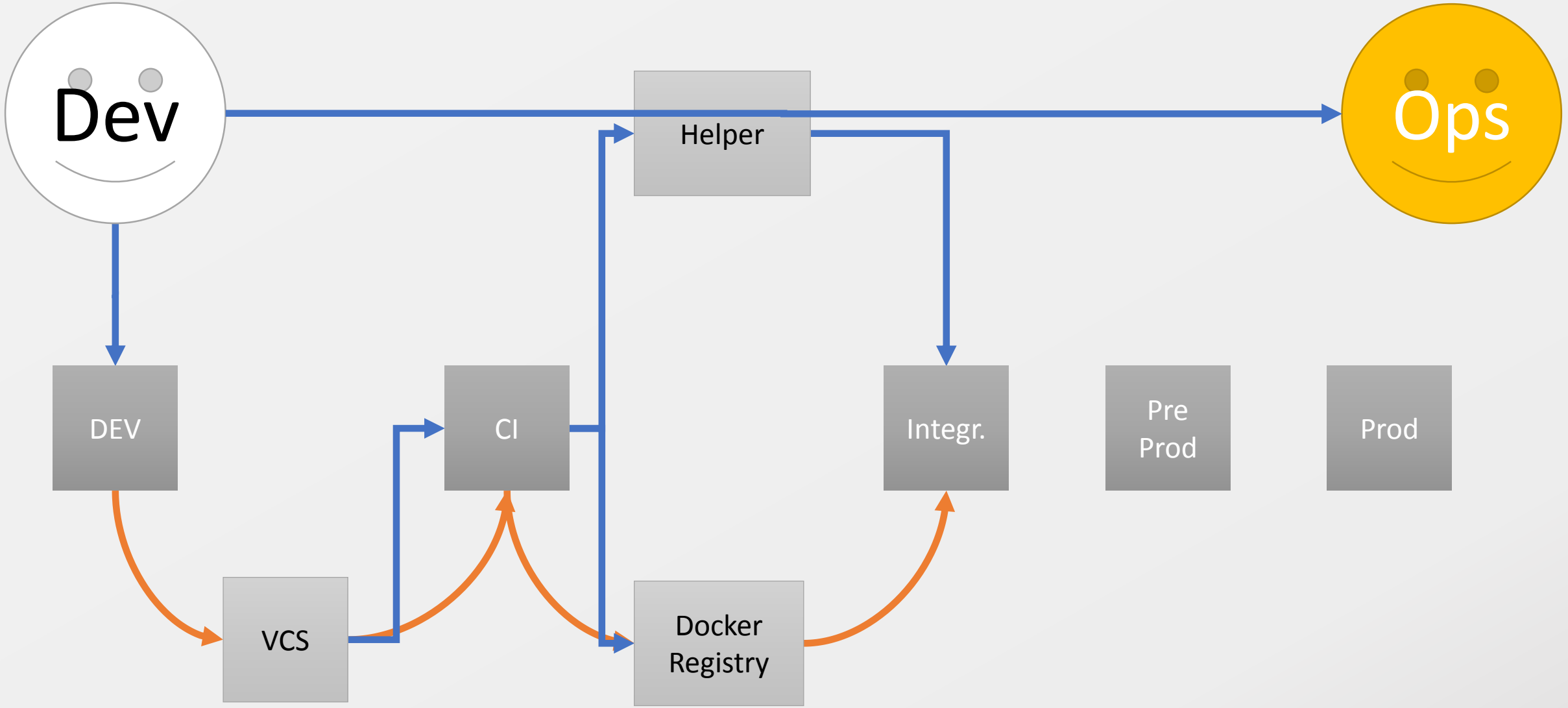
orange lines are data flow and **blue lines are control flow**


Dev
CI
Integration
Pre-Prod
Prod

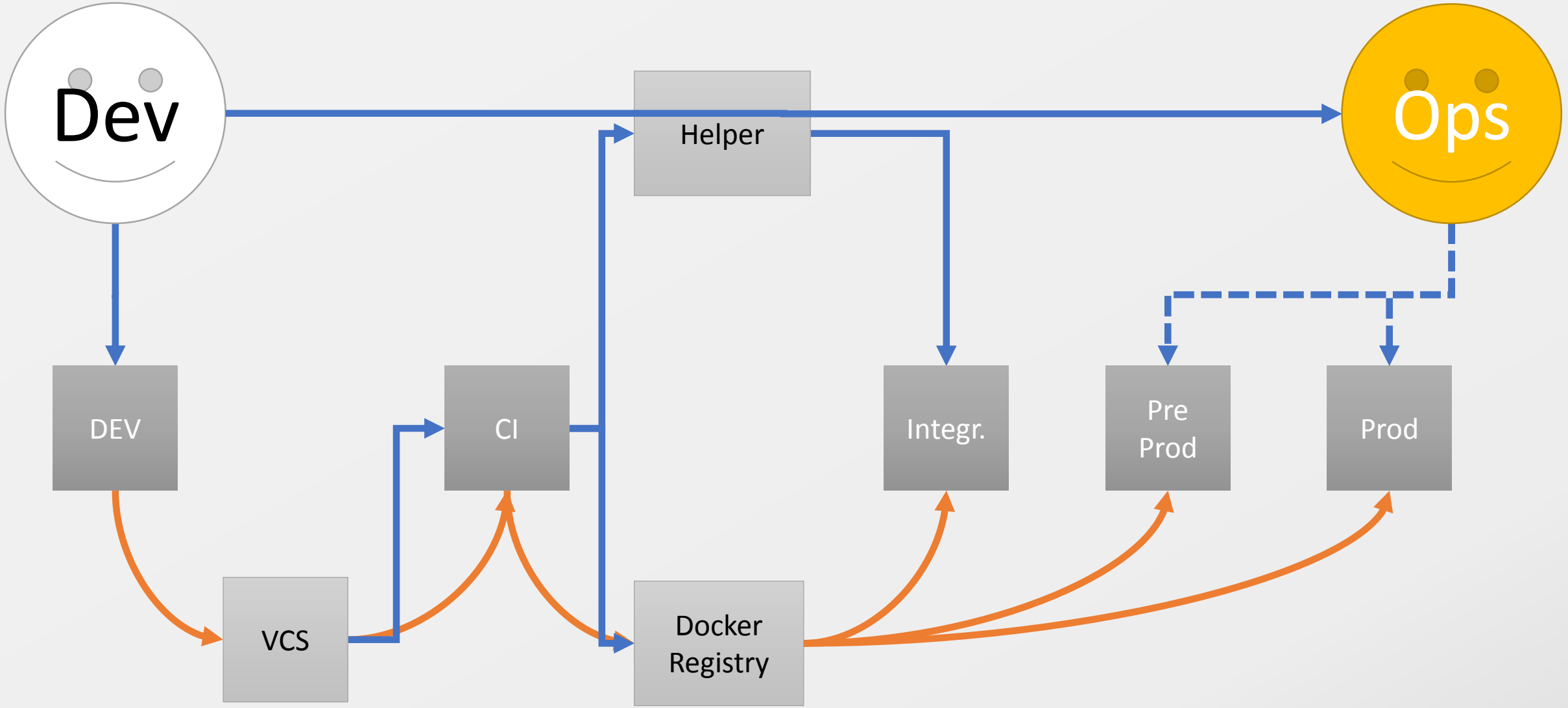

Dev
CI
Integration
Pre-Prod
Prod



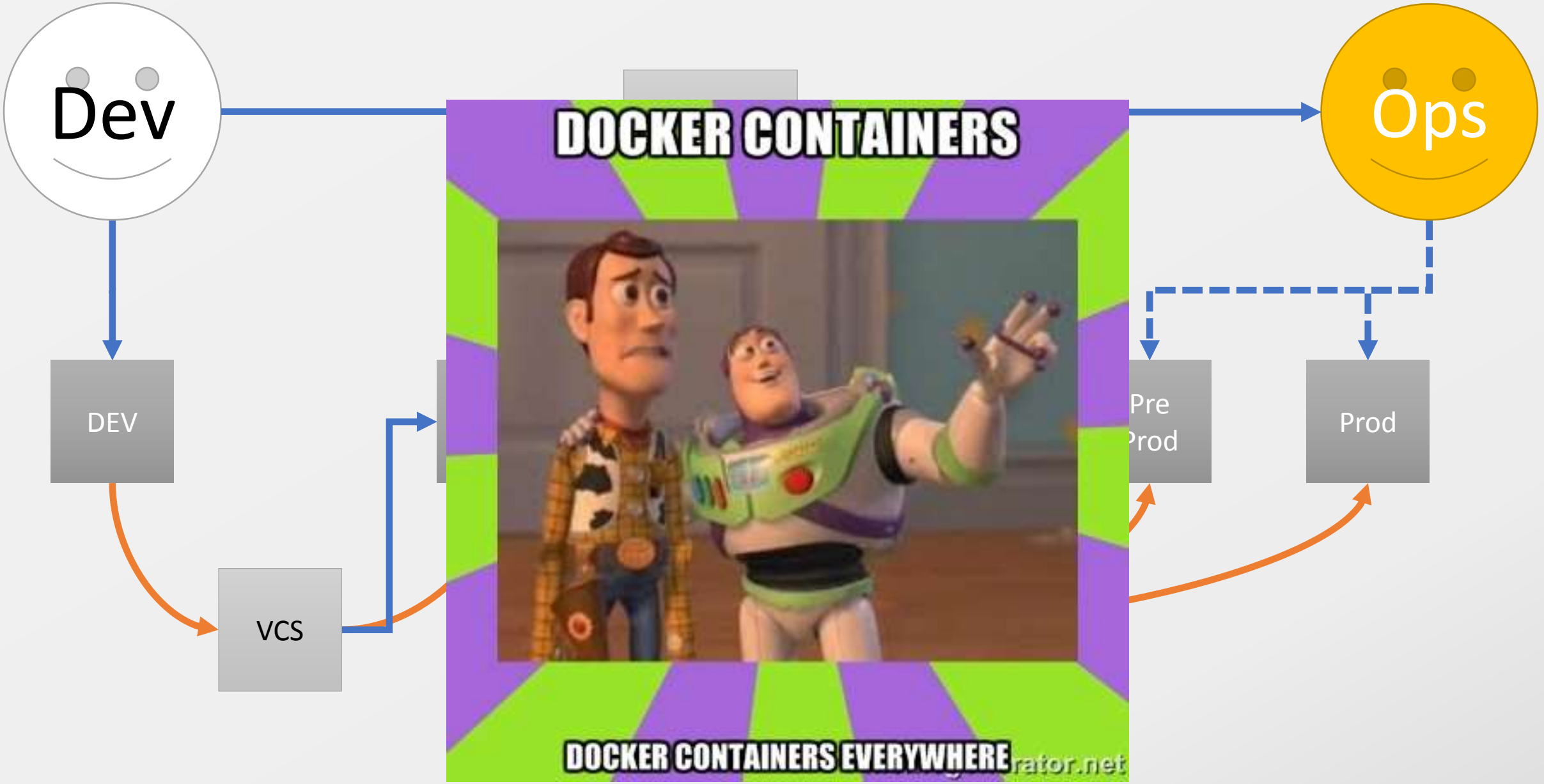
orange lines are data flow and **blue lines are control flow**



orange lines are data flow and **blue lines are control flow**



orange lines are data flow and **blue lines are control flow**



orange lines are data flow and blue lines are control flow

Easy Packaging

Consistent Dependencies

Repeatable Deployments

Easy Automation

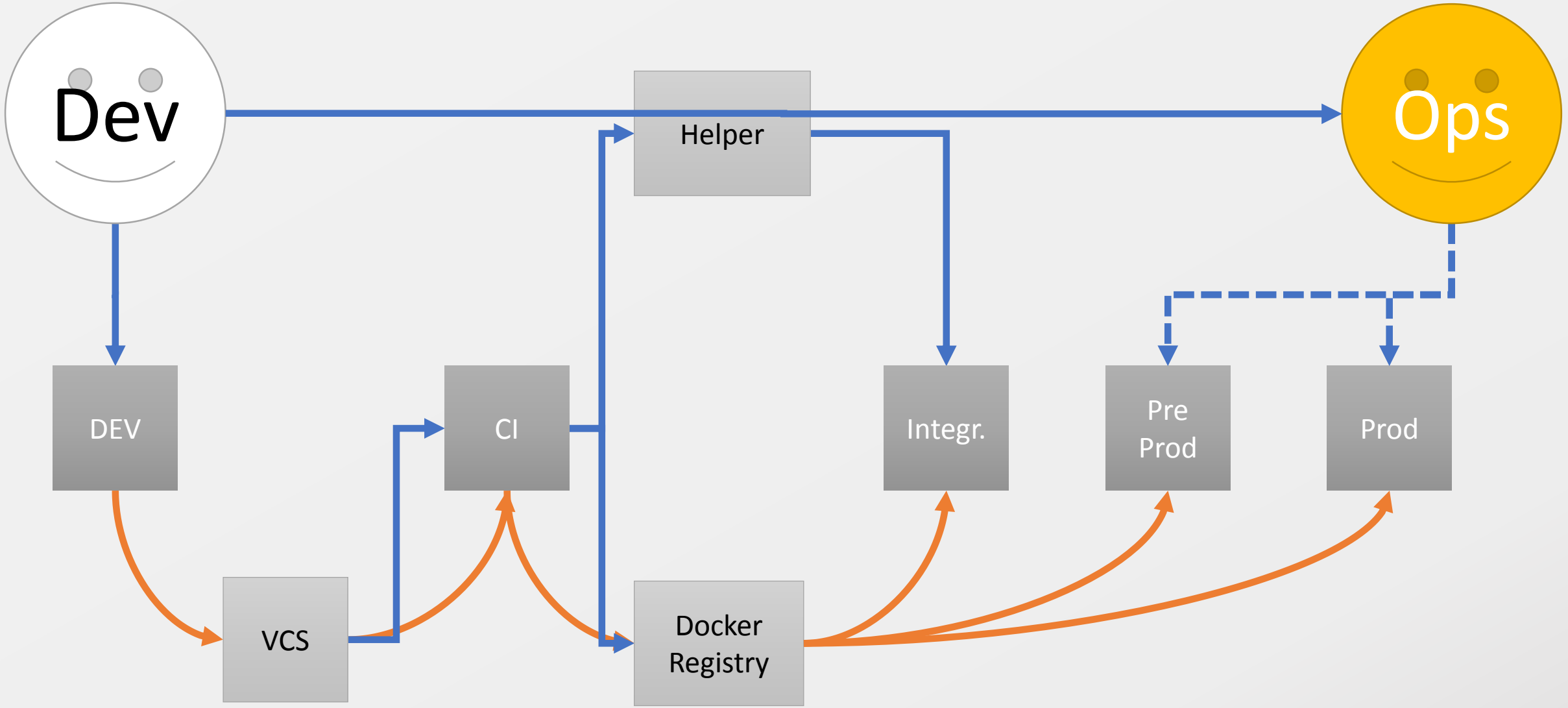
Continuous Delivery

Logging

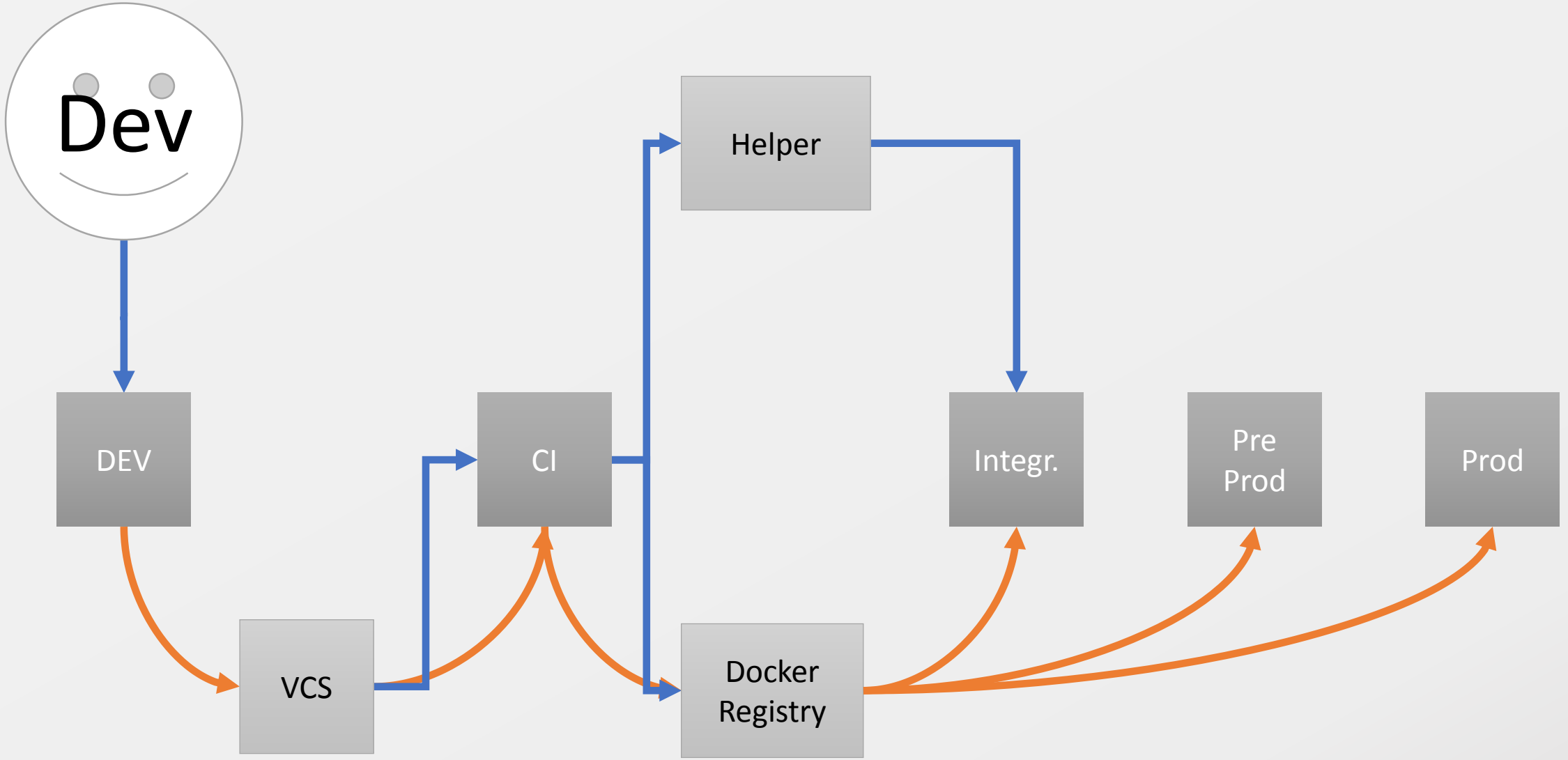
Scaling

Monitoring

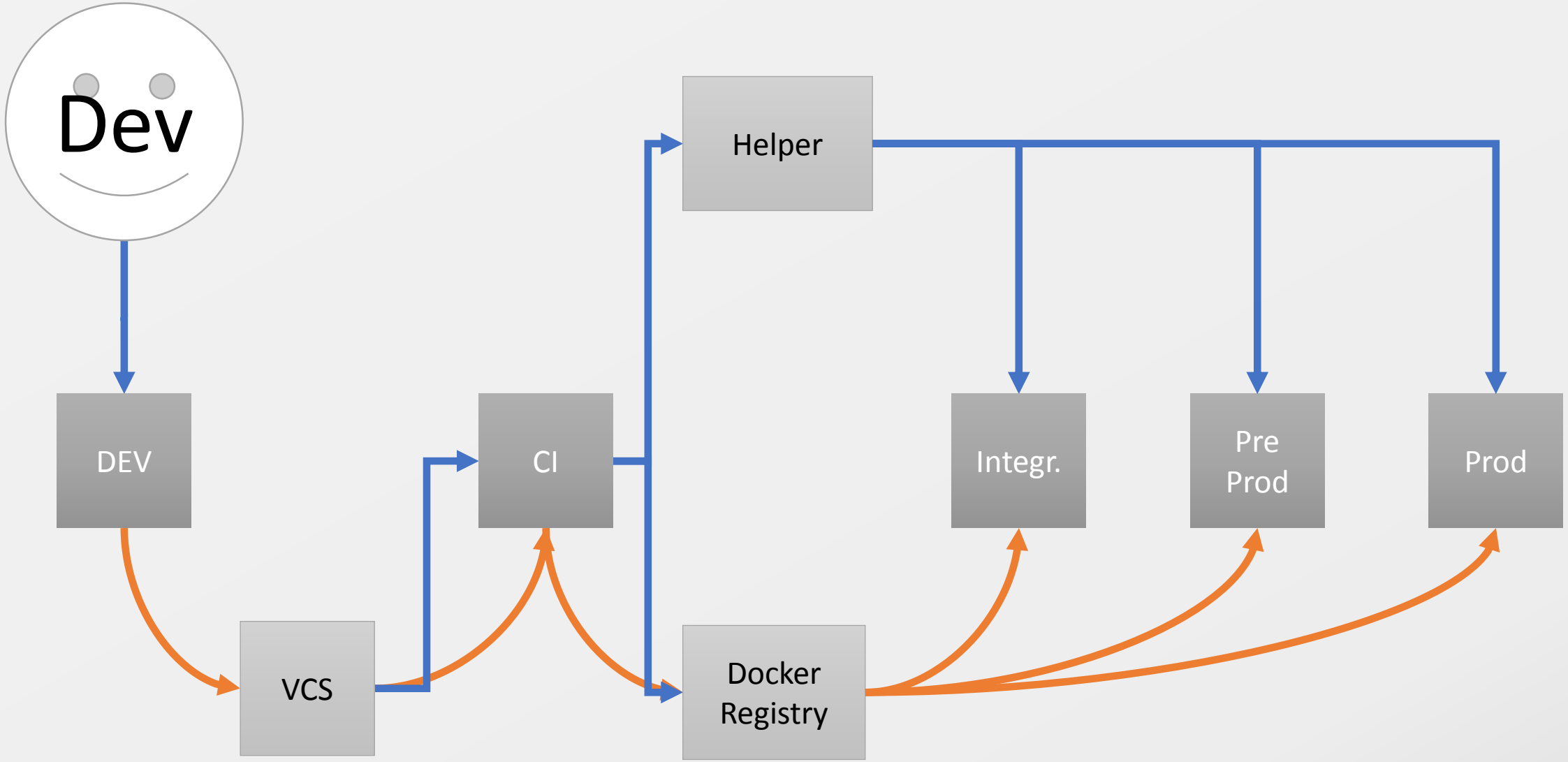
Automatic Recovery



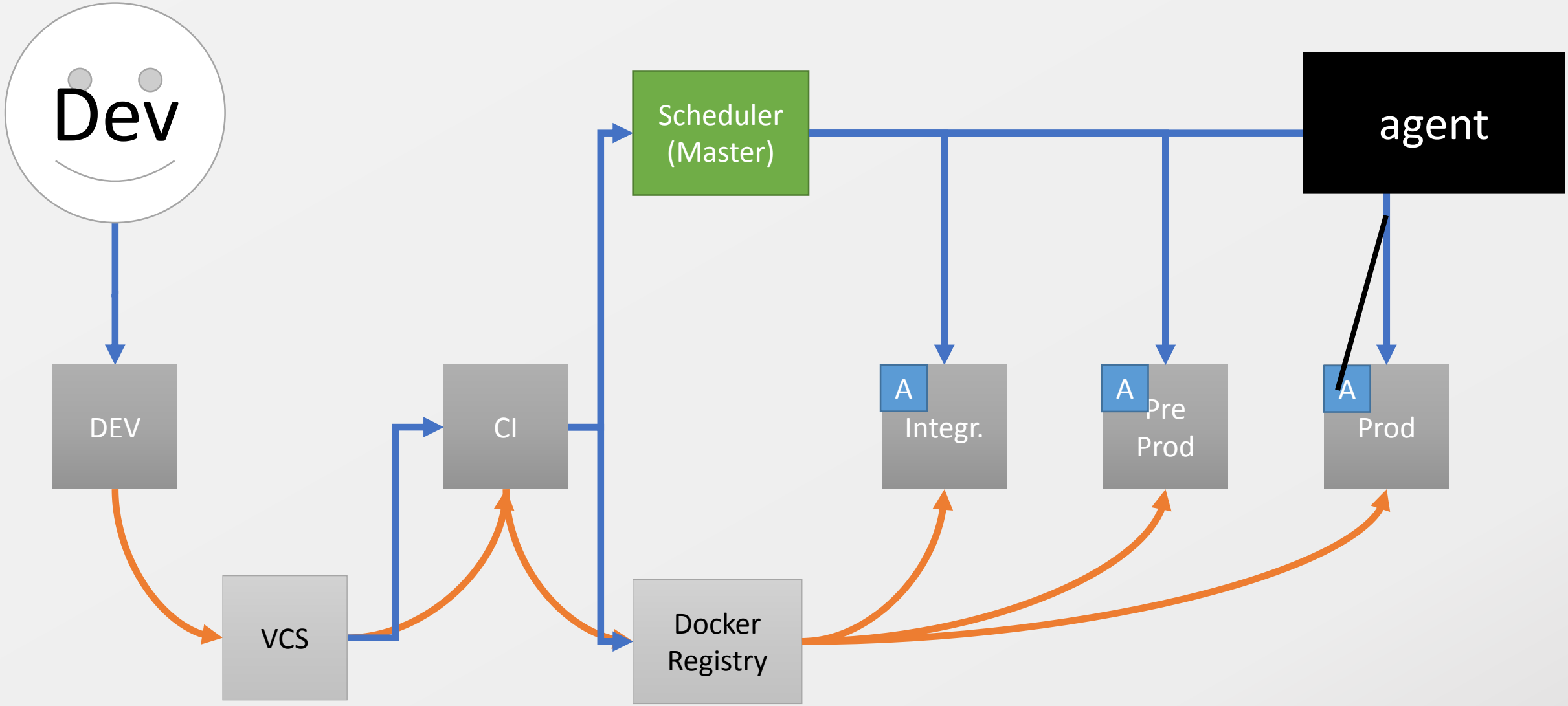
orange lines are data flow and **blue lines are control flow**



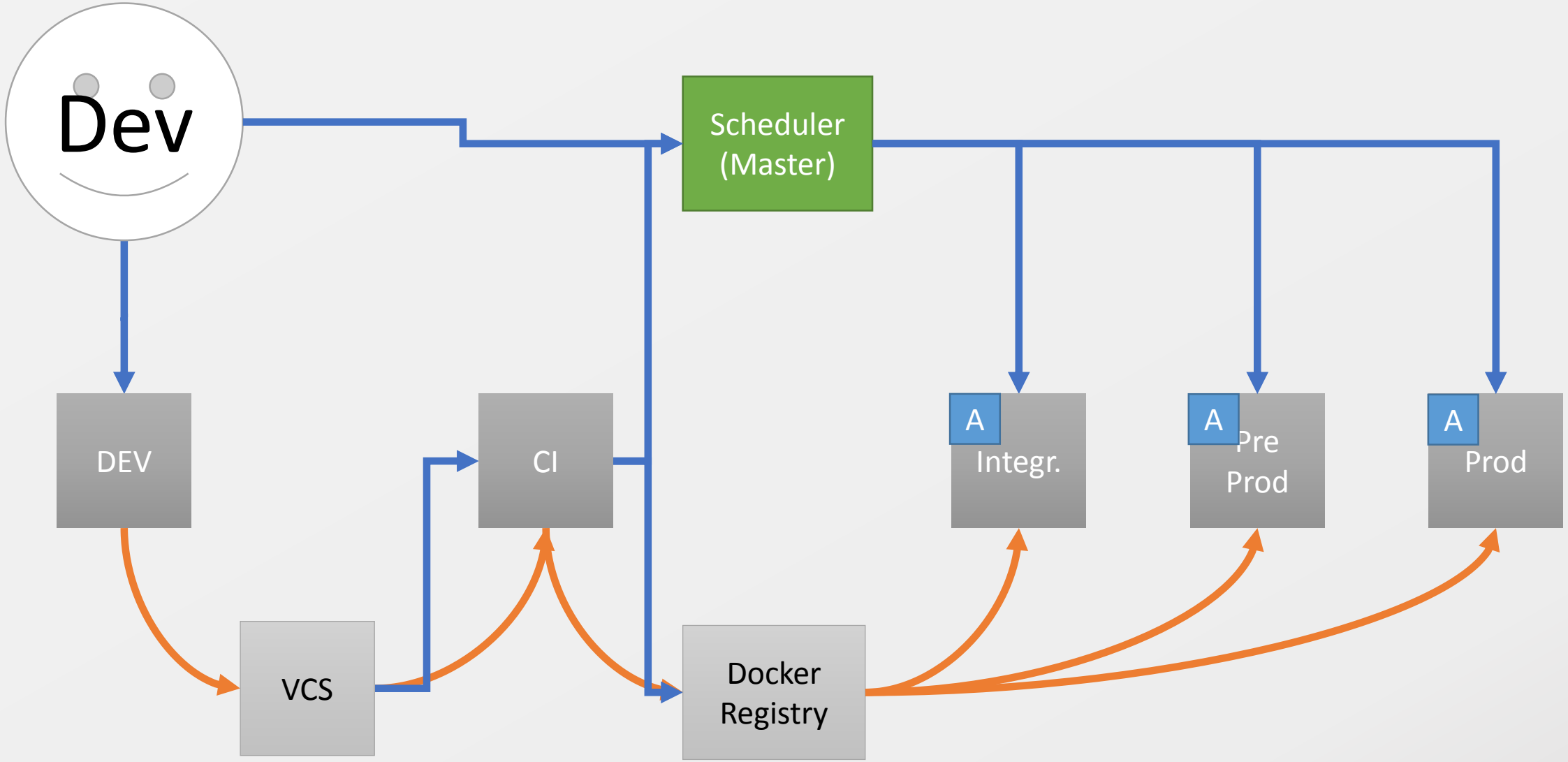
orange lines are data flow and **blue lines are control flow**



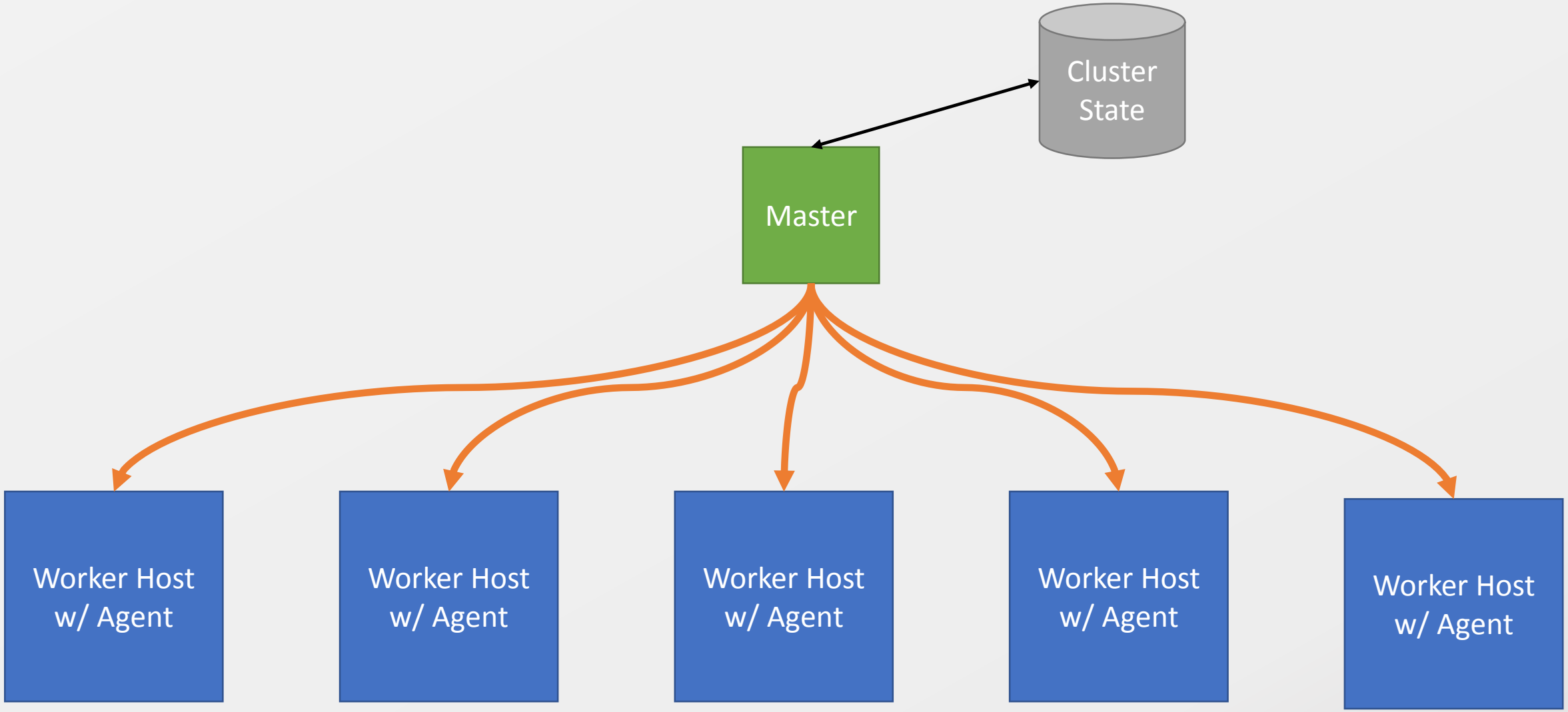
orange lines are data flow and **blue lines are control flow**

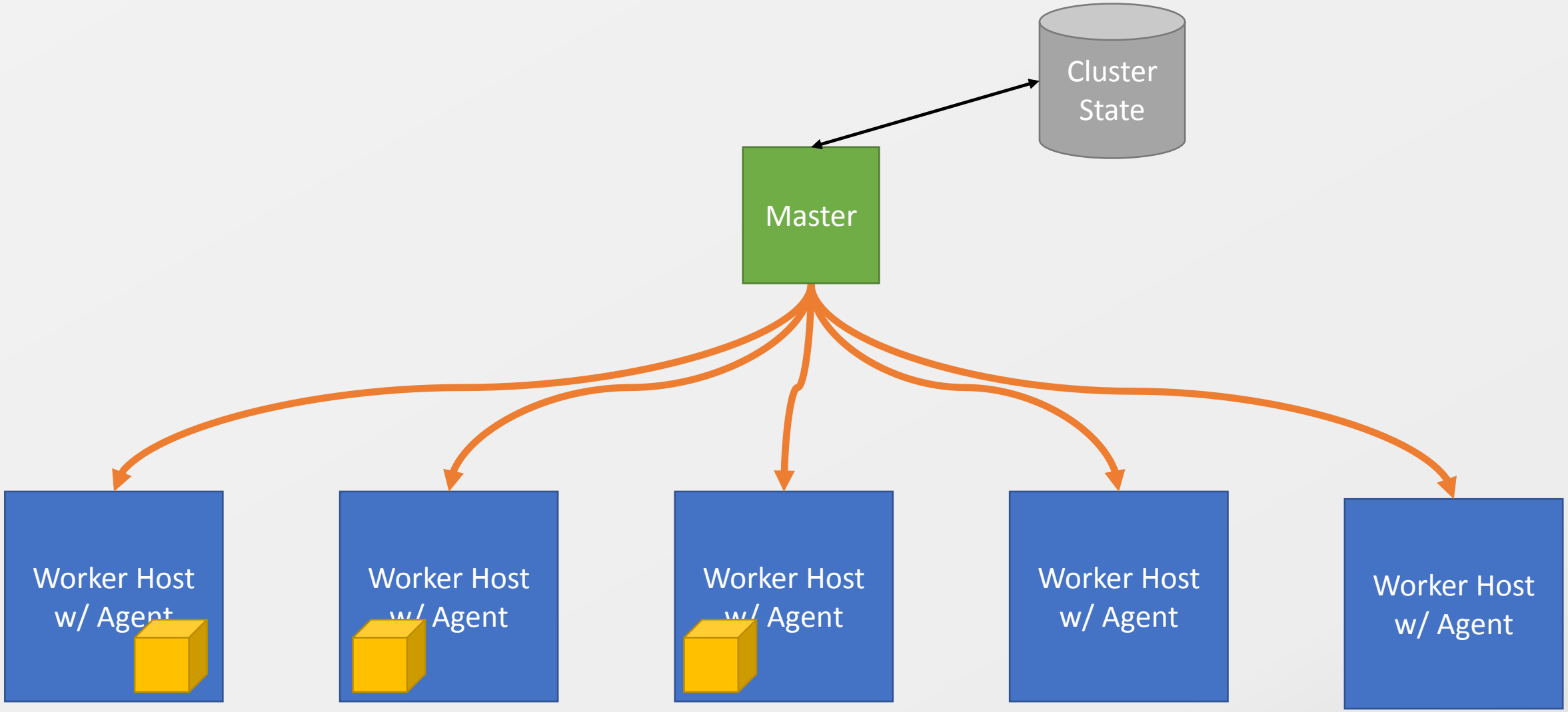


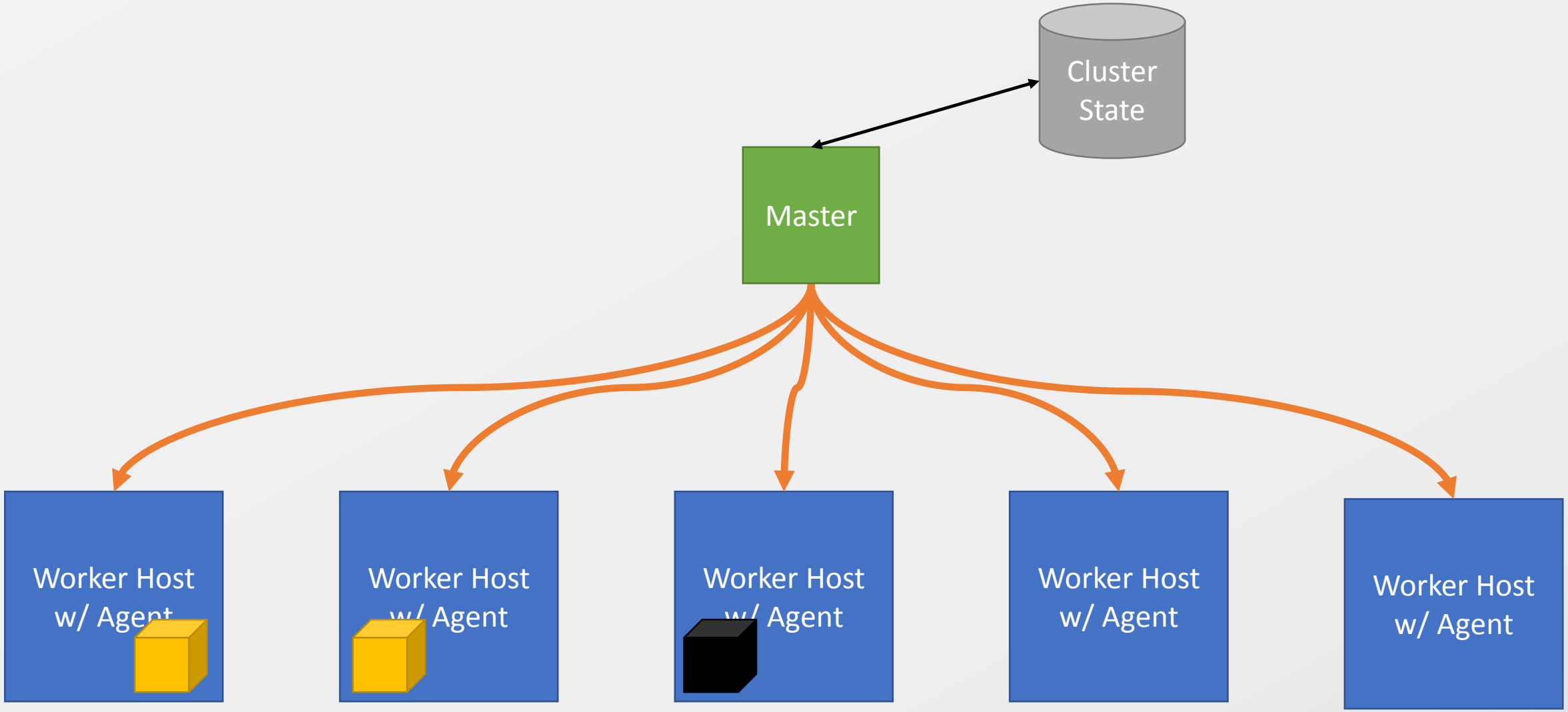
orange lines are data flow and **blue lines are control flow**



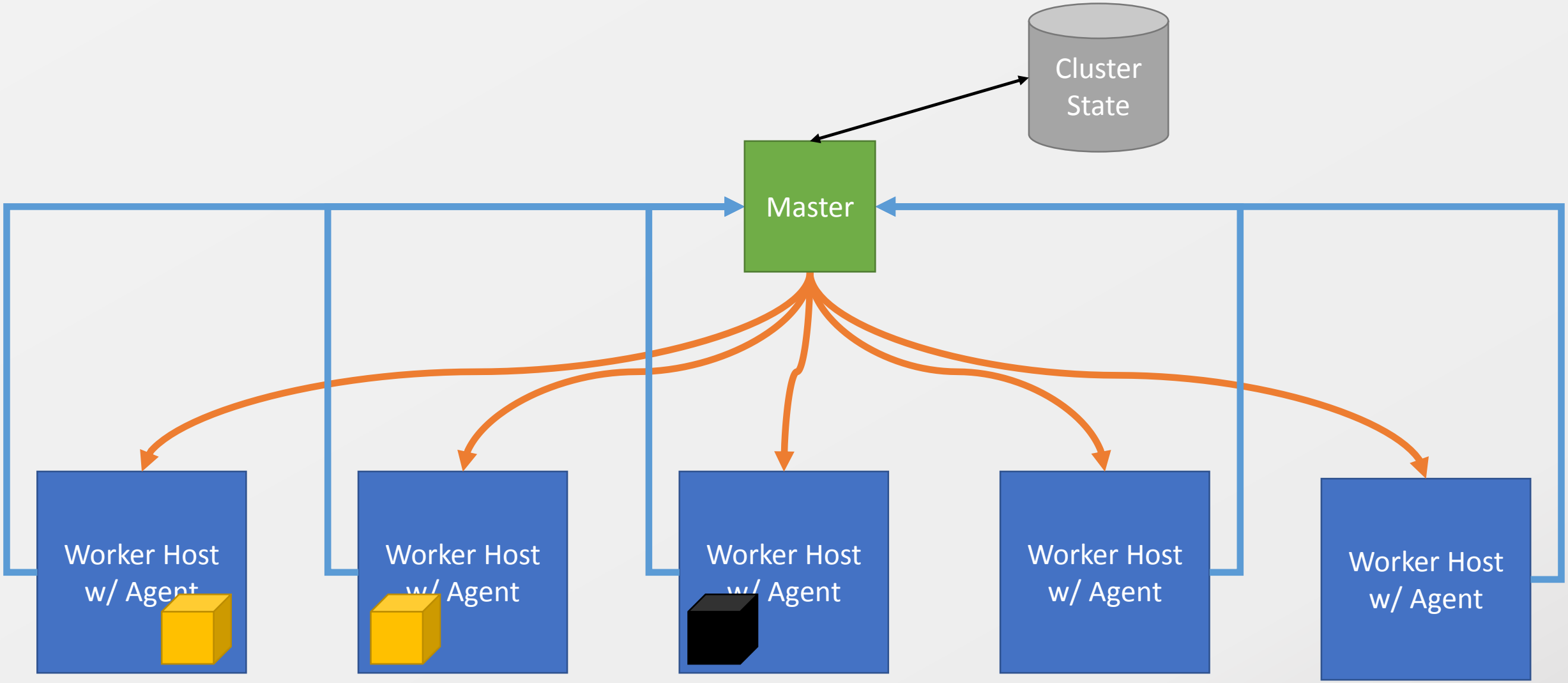
orange lines are data flow and **blue lines are control flow**


Cluster State
Master
Worker Host w/ Agent
Worker Host w/ Agent
Worker Host w/ Agent
Worker Host w/ Agent
Worker Host w/ Agent

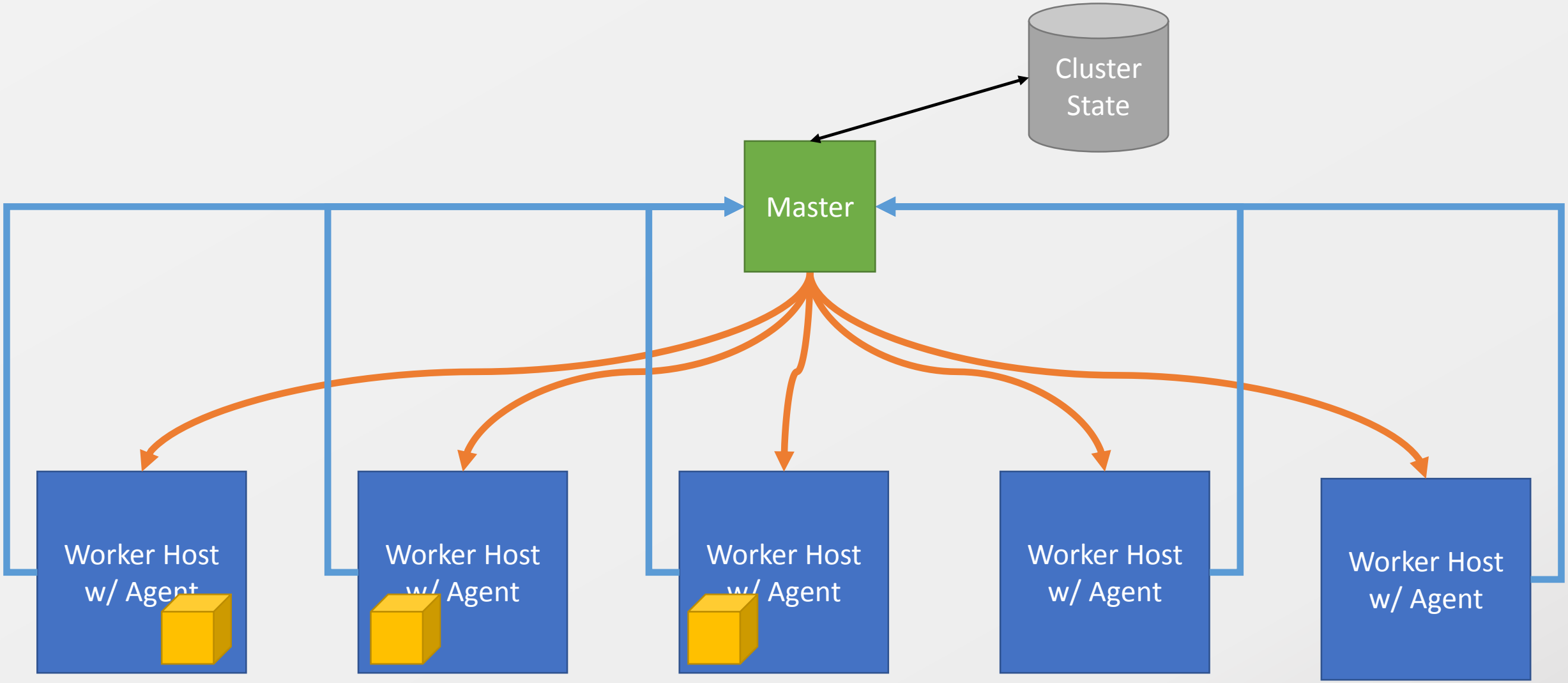

Cluster State
Master
Worker Host w/ Agent
Worker Host w/ Agent
Worker Host w/ Agent
Worker Host w/ Agent
Worker Host w/ Agent



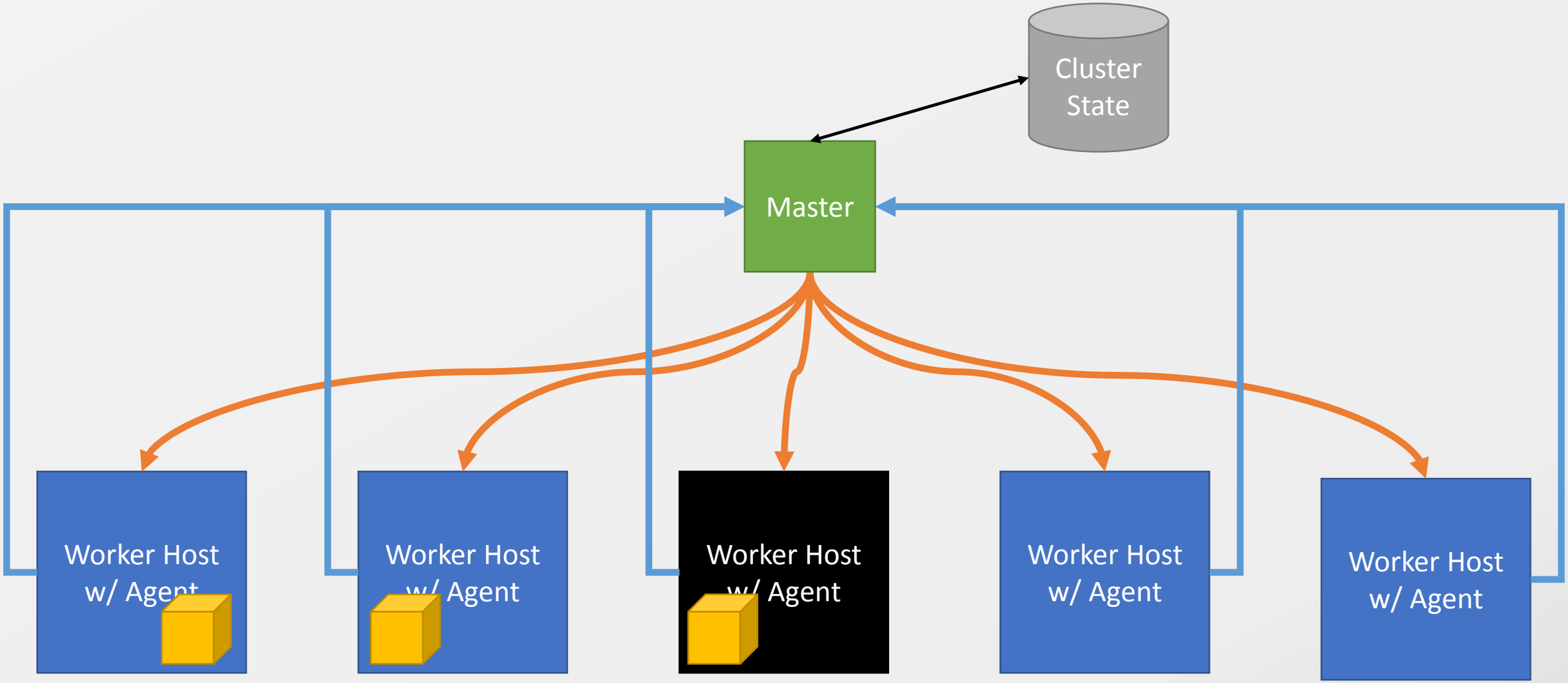
commands desired state vs. **reports actual state**



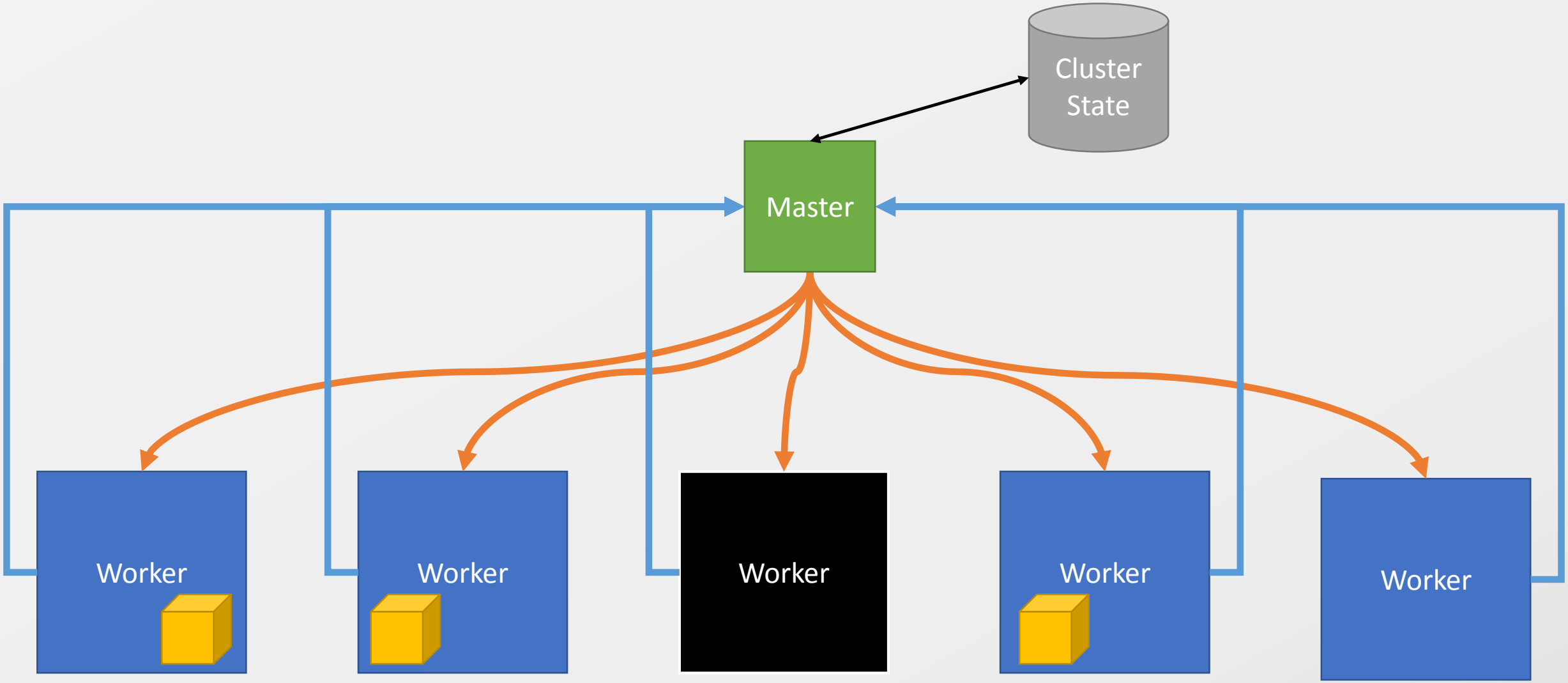
commands desired state vs. **reports actual state**



commands desired state vs. **reports actual state**



commands desired state vs. **reports actual state**



commands desired state vs. **reports actual state**

Orchestration

Which container runs on which host?

Orchestration

How does the traffic come there?

Which container runs on which host?

Orchestration

How does the traffic come there?

Which container runs on which host?

Orchestration

How do the services talk with each other?

How does the traffic come there?

Which container runs on which host?

Orchestration

How do the services talk with each other?

How to do updates w/o service interruption?

How does the traffic come there?

Which container runs on which host?

Orchestration

How do the services talk with each other?

How to do updates w/o service interruption?

How to scale out?

How does the traffic come there?

Which container runs on which host?

Where do the logs go?

Orchestration

How do the services talk with each other?

How to do updates w/o service interruption?

How to scale out?

How does the traffic come there?

Which container runs on which host?

Where do the logs go?

Orchestration

How do the services talk with each other?

How to do updates w/o service interruption?

How to scale out?

Spotify Helios

Docker Swarm

Kubernetes

Orchestration

Rancher

Newrelic Centurion

Apache Mesos with Marathon

Spotify Helios

Docker Swarm

Kubernetes

Orchestration

Rancher

Newrelic Centurion

Apache Mesos with Marathon

Google Container Engine

Redhat Openstack

Cloudfoundry

Cloud Provider

Microsoft Azure

Amazon EC2 Container Service

heroku


Dev
CI
Integration
Pre-Prod
Prod

The Twelve-Factor App

Introduction

In the modern era, software is commonly delivered as a service: called *web apps*, or *software-as-a-service*. The twelve-factor app is a methodology for building software-as-a-service apps that:

http://12factor.net

I. Codebase

One codebase tracked in revision control, many deploys

II. Dependencies

Explicitly declare and isolate dependencies

III. Config

Store config in the environment

IV. Backing Services

Treat backing services as attached resources

V. Build, release, run

Strictly separate build and run stages

VI. Processes

Execute the app as one or more stateless processes

VII. Port binding

Export services via port binding

VIII. Concurrency

Scale out via the process model

IX. Disposability

Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity

Keep development, staging, and production as similar as possible

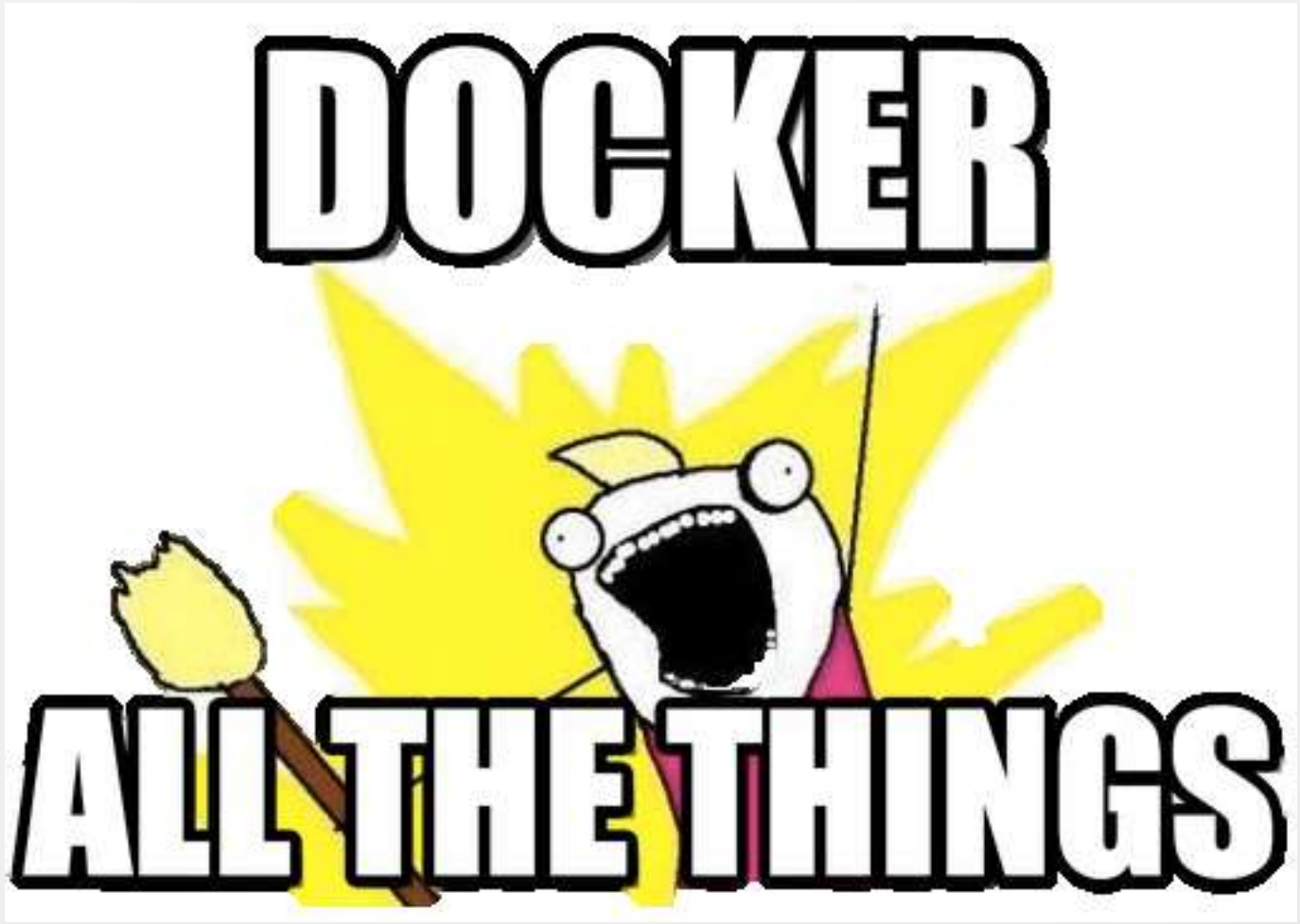
XI. Logs

Treat logs as event streams

XII. Admin processes

Run admin/management tasks as one-off processes


Dev
CI
Integration
Pre-Prod
Prod

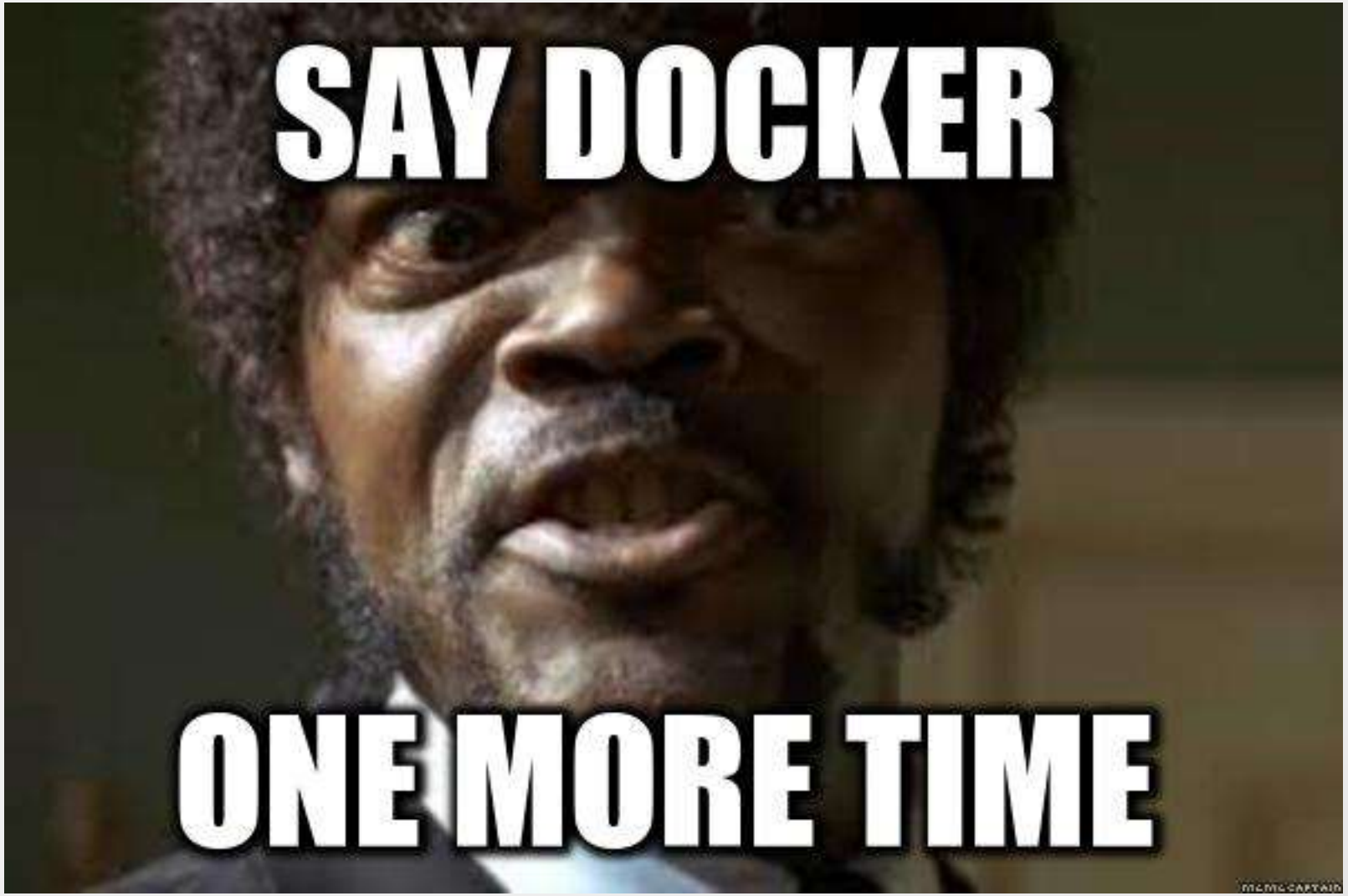

DOCKER
ALL THE THINGS

Databases

- Very own requirements:
  - State-full
  - Low CPU but high I/O
  - Distributed data
  - (usually) complex to scale
  - (usually) not built for “hot swap”
- → **not 12 Factor applications**

Databases

- Don’t dockerize databases
- Keep them external to the system
  - DBA will ~~love you~~ not hate you
  - Easier to maintain
  - Very own cluster mechanisms
  - Easier routing in Datacenters


SAY DOCKER
ONE MORE TIME