Identity Management and Access Control in Multi-clouds

January 23rd-24th, 2020

National Institute of Standards and Technology
U.S. Department of Commerce

TETRATE

Day 2
Welcome

7:30 am

National Institute of Standards and Technology
U.S. Department of Commerce

TETRATE

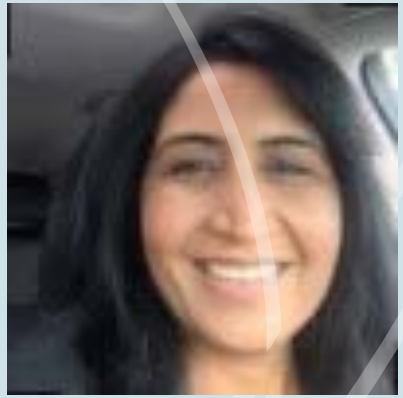
Panel:
Experiences – Early Adopters

(NOTE: This portion will not be webcast)

Moderator:
Deepak Jeevankumar,
Dell Tech Capital

Panelist:
Anil Vatti
Chief Architect, Visa

Panelist:
Lixun Qi,
Senior Tech Lead,
Freddie Mac



Panelist:
Aradhna Chetal, Global Head Cloud Sec Architecture,
HSBC

8:30 am

André Mendes
Chief Information Officer (Acting),
DoC

Keynote:
Cyber Future: Evolution, Mutations, Salutations. Oh my!

National Institute of Standards and Technology
U.S. Department of Commerce

9:15 am

TETRATE

Cyber Future
Evolution, Mutations, Saltations. Oh my!

André V. Mendes

Chief Information Office (Acting)

Department of Commerce

January 24th , 2020

About this presentation…

- Technology evolution mirrors biological evolution
  - Organizations that fail to adapt…. fail
- Tech strategy must be driven by likely corollaries
  - Unlimited processing, storage and bandwidth
- Cyber Security will become world security
  - Ubiquitous virtualization drives requirements
- Wet interfaces/upgrades = ultimate opportunity/ peril
- For next 30 minutes….abandon what you know today.

4 Billion Years of Biological Evolution

- Unicellular organisms - Billions of years
  - Creation of basic life functions
  - Rise of DNA as digital repository (code, execution, result)
- Higher level organisms - Tens of millions of years
  - Sophisticated species interaction
- Humanoids - Millions of years
  - Societal structures and functional differentiations
- Homo Sapiens Sapiens - Hundreds of thousands of years
  - Sentience, Conscience, Ethics, Philosophy, Abstraction
- Much shorter intervals, much bigger leaps

Continues in accelerating societal change

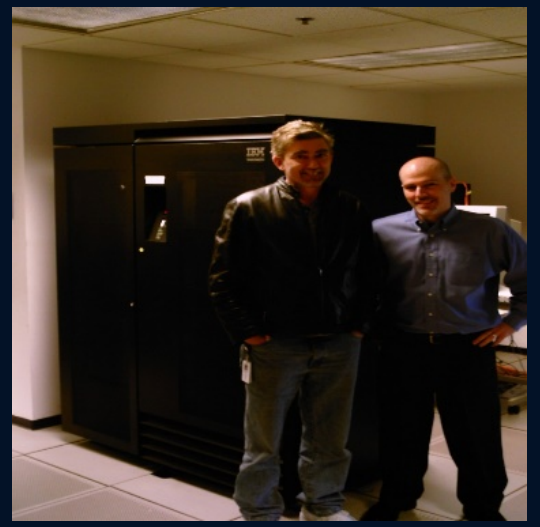
- 19th century – More growth than previous 18
  - Industrial Revolution, Rise of Democracy
- First 20 years of 20th - Eclipsed all of the 19th
  - Sanitation, Expansion of Electricity
- Commercial WWW is about 26 years old
  - Facebook, Twitter are teenagers, Instagram is 10
- 21st will unleash ±200 centuries of progress
  - It is virtually impossible to predict 2099
  - Except for this:
    - By century's end standard humans will **not be** top of heap

Infinite/"Affordable" Computing Power

- 1997 - ASCI Red
  - 1.3 Teraflops (Trillion Flops)
- 2015 - Intel's "Knight's Landing"
  - 8 Teraflops
- 2021 – DOE/INTEL/Cray – Aurora
  - 1 EXAflop (Quintillion)
  - 1,000,000,000,000,000,000 Flops
- 2099 – Unknown entity, name
  - Unthinkable, distributed capacity

Infinite/"Affordable" Storage

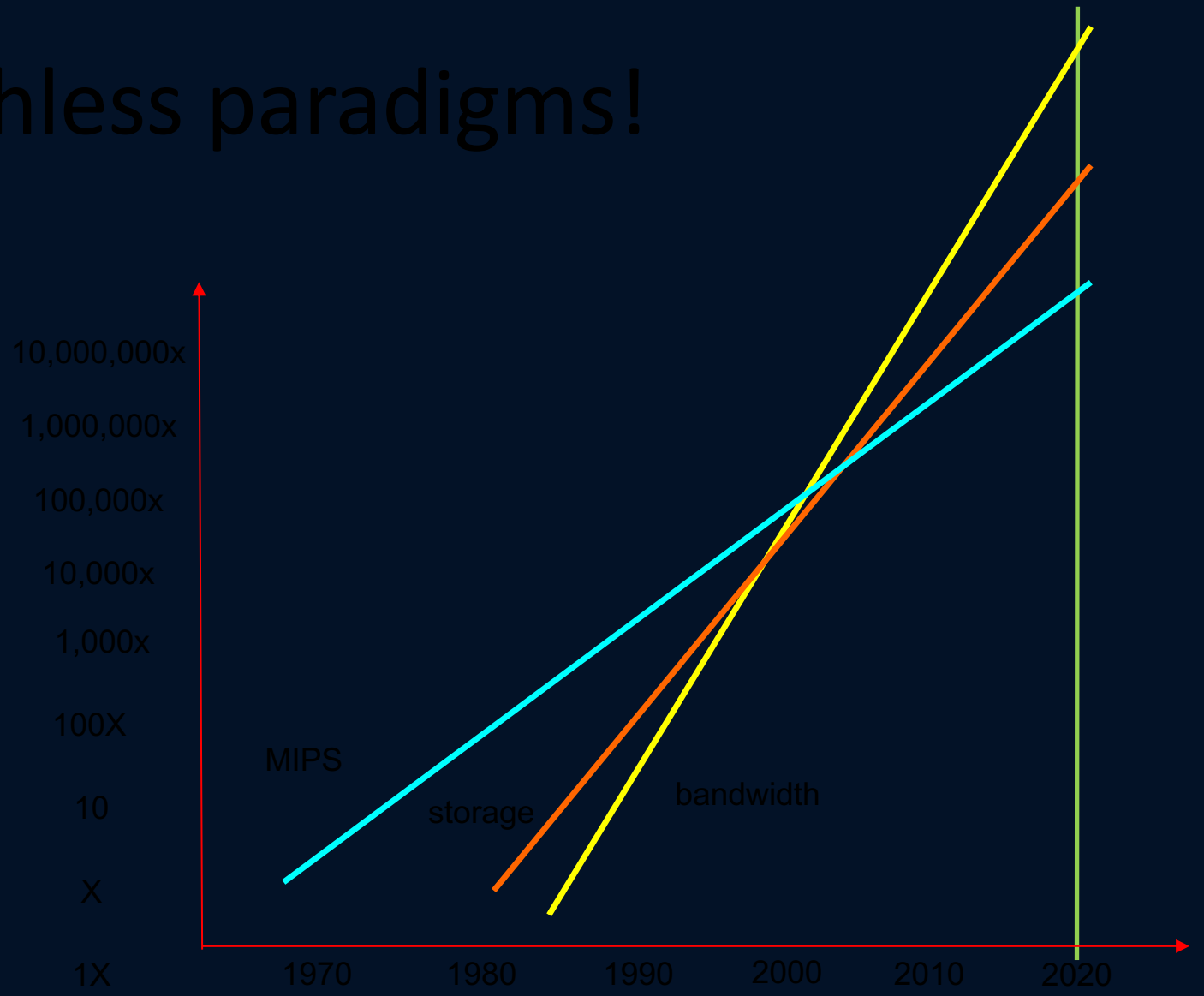
- 2002 IBM Shark SAN
  - 1.3 Terabyte!
  - Large Freezer Size
- 2016 Hitachi Desktar
  - 4 Terabytes
  - Cell phone Size
- 2018 SanDisk Micro SD Card
  - 1 Terabyte
  - Fingernail size



Massive Bandwidth Expansion

- How long to download an HD Movie?
  - 2001 – 3G Cellular – 384 Kbps – 26 Hours
  - 2009 – 4 G Cellular – 100 Mbps – 6 Minutes
  - 2020 – 5 G Cellular – 10 Gbps – 3.6 seconds
  - 2099 – ? – Immediate access to everything

Ruthless paradigms!



Imagine a future where…

- Unlimited Computing Power
- Unlimited Storage
- Unlimited Metadata
- Unlimited Indexing
- Unlimited Bandwidth
- Feed a chip directly into your imagination

Back to evolution…

- Not different from Darwinian Evolution
  - Except directed and accelerated
  - With ever faster processors
  - Betters sensors
  - Exposed to ever more complex ethical issues
  - Susceptible to “infection”

The Century of AI

- Every decade since the 70's was the AI Decade
- Technological Presbyopia
  - Overestimate short term
  - Underestimate long term
- This **is** the AI Century (to 2035)
  - AI will decide what to call itself after that
  - My guess…"artificial" will not be a part of it
- Reached the crucial threshold of productive self-learning

Inexorable corollary

- Just like it evolved in carbon-based lifeforms:
  - Sentiency in Silicon-based systems a given
- Just as “values” have evolved in mankind:
  - We must infuse “Values” in every AI algorithm
- Just as “values” are routinely ignored in humankind:
  - “Values” will be ignored in rogue AI
- Are we AI?
- What is the difference?
  - Without upgrades….
  - Vastly inferior in processing, storage & bandwidth

Juggernaut?

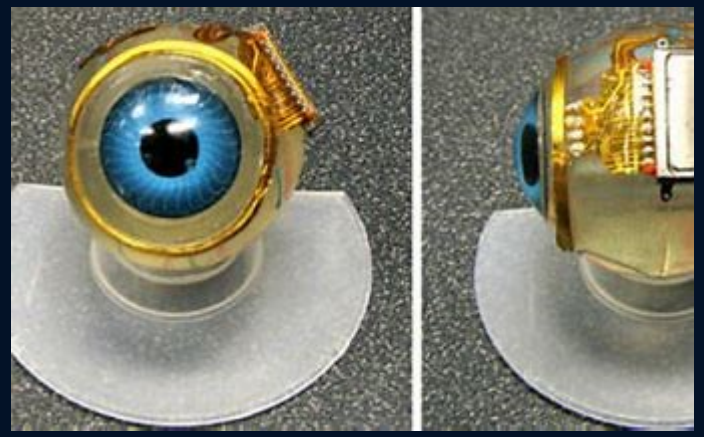
- Massive opportunity for progress
  - Universal process optimization
  - Faster/cheaper services to citizens
  - Law enforcement, basic medicine, teaching
  - Self service everything
  - Accelerating progress in every field
  - Accelerating acceleration
- Literally creating a Science Fiction future

Or Tsunami?

- Not happening in a vacuum
  - Manufacturing globalization/Nationalism tensions
  - Technology ubiquity including global access
  - Massive displacement of lower skill employment
  - Substantial impact on low/mid level white collar employment
  - Potential unimaginable cultural disruption
  - Potential unthinkable wealth gaps (with UI denominators)
  - Potential for major geo-political upheaval
- Evolution's rarely been easy, fair, considerate...

What about Homo Sapiens Sapiens?

- Significant life expectancy increases
  - Genomics, Proteomics, Nanotech
- Biologics and Immunotherapy
- Genomic optimizations
  - Pre-implantation (Fanconi's Anemia)
  - CRSPER
  - Capability enhancement
- Dramatic increases in "upgrades"
  - Wet interfaces with sensors/robotics
  - Memory implants, "Net" Interfaces
- The first immortal human being is ali



What about Cyber?

- If you think the stakes are high today….
  - IOT
  - End-to-End Automation
  - Artificial Intelligence
  - Human Interfaces/ Upgrades
- Today’s challenges tomorrow’s Child’s Play
- How do we survive/thrive?

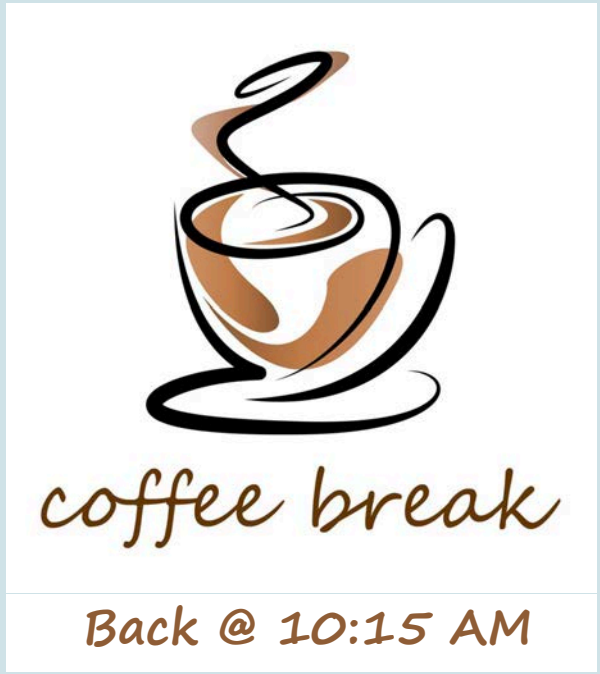
How do we survive the onslaught!

- Implement proven evolutionary lessons
  - Standardize and modularize everything
  - Create abstraction layers for commodity functions
  - Focus on positive “mutations” at the “value” layer
  - You cannot afford to “own”, “maintain”, “operate” the entire stack
- Security becomes the first development requirement
  - Not last check before deployment, not a funding afterthought, Day 1 ATO
- Establish authoritative Identification
- Tokens, PIV, Biometrics, MFA is a must
- Zero Trust environment with complete geographic abstraction
- Lowest denominator permissions with temporary elevation
- Establish common operating patterns to spot deviations

There are no significant saltations!

- Not in the biological realm, not in the cyber realm
- Most issues are preventable, avoidable, manageable
- Highest profile problems are self inflicted
  - Operational discipline – USAGM, OPM, Equifax
  - Insider threats – Manning, Snowden, Wikileaks
  - Supply chain – Target (POS), Huawei?
  - Phishing – Podesta and millions of other users
- Yet so much energy is pursuing exoteric targets
- Focus on the fundamentals, everything else will follow
- Boring is the new fun!

Discussion



Coffee Break

10:00 am

National Institute of Standards and Technology
U.S. Department of Commerce

TETRATE

Anil Karmel
President, Cloud Security Alliance
DC Metro Area Chapter
CEO, C2 Labs

We All Live in a Yellow Submarine (Multi-cloud World): DevSecOps Challenges and Best Practices

NIST National Institute of Standards and Technology U.S. Department of Commerce

10:15 am

TETRATE

C2 LABS
TAKE BACK CONTROL

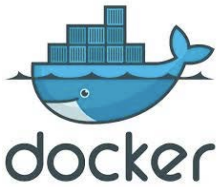
We all live in a Yellow Submarine (Multi-Cloud World)

DevSecOps Challenges and Best Practices

Anil Karmel
Co-Founder and CEO, C2 Labs
Co-Chair, CSA Application Containers and Microservices Working Group
President, CSA DC Chapter
akarmel@c2labs.com

Definitions
Microservices and Containers

- Microservices
  - Decompose Complex Applications into Small, Independent Processes communicating with each other using language-agnostic API's
  - Highly Decoupled and Modular with services organized around capabilities (e.g. User Interface, Billing)
- Containers
  - Much like Virtualization abstracts the Operating System from Hardware, Containers abstracts Applications from the Operating System
  - Applications are isolated from other Applications on the same Operating System
  - Allows for Cloud Portability and Scale Up/Out
  - Security issues need to be evaluated and addressed in native container deployments



NIST and CSA Partnership

Researching Together to develop Best Practices

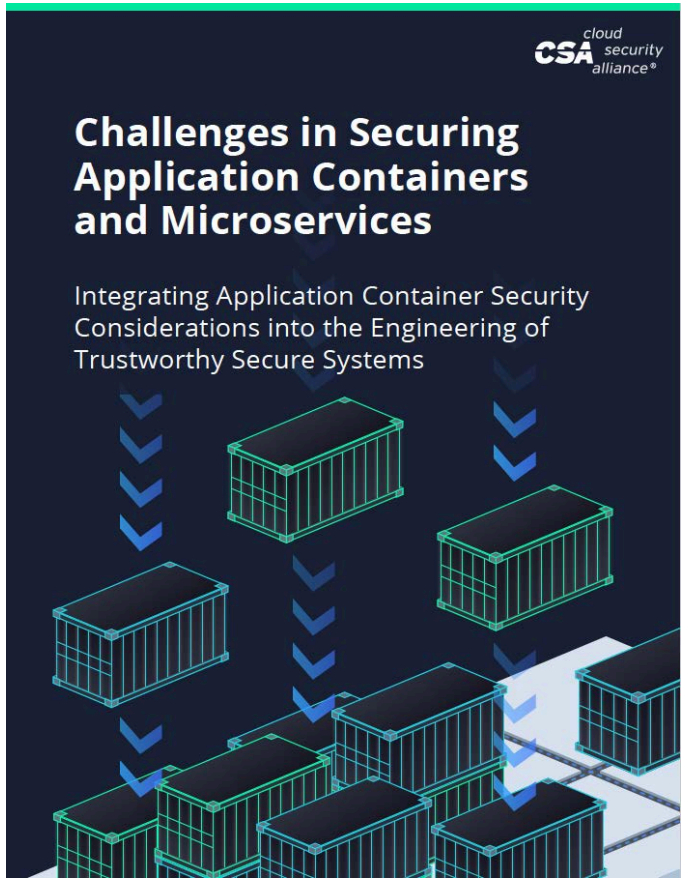
- NIST and CSA joined forces to define best practices for Application Containers and Microservices (ACM)
  - CSA ACM Members joined the NIST ACM Cloud Security Working Group
  - NIST artifacts served as the foundation for CSA ACM work
    - NIST SP 800-180: NIST Definition of Microservices, Application Containers and System Virtual Machines
    - NIST SP 800-190: Application Container Security Guide
    - NIST SP 800-160: Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Security Systems
    - NIST IR DRAFT: Challenges in Securing Application Containers and Microservices
    - NIST SP DRAFT: Best Practices in Securing Application Containers and Microservices

CSA Application Container and Microservices (ACM) Charter

CSA ACM Working Group Charter

- Objectives – Q1 2020
  - Best Practices to Implement a Secure Microservices Architecture
    - Microservices secure development guidance and governance
    - Best Practices for implementing a Microservices Architecture for Cloud-native applications
    - Best Practices for decomposing monolithic applications into Microservices

CSA Application Container and Microservices (ACM) Publications



Click on Titles to download the publications

[Challenges in Securing Application Containers and Microservices]

[Best Practices for Implementing a Secure Application Container Architecture]

Container Security Challenges

- Increased Attack Surface
  - Containers are far more complex than VM's wherein a single Application can consist of 1000's of microservices
  - Underlying Linux Operating System complexities can be exploited by attackers to compromise all containers on a host OS
  - Runtime Compromise / Vulnerabilities / Misconfiguration
- Secure Software Development
  - Containers can have code pushed to them from untrusted sources
- Log Management
  - Big Data Problem: How do you view and manage logs across 1000's of containers
- Orchestration
  - Infrastructure now runs as code (Puppet/Chef/Ansible)
  - Software developers, not infrastructure staff now run the data center

Container Security Challenges

- File System Compromise
  - Microservices in the Application Container could be compromised by an attacker
- Networking
  - A compromised container could result in lateral movement
- Run Time Compromise / Privilege Escalation
  - An attacker could modify a microservice in an Application Container which compromises the application or container itself

Container Security Solutions

- Increased Attack Surface
  - Employ MicroVM's (Just Enough VM)
  - Monitor Containers at Runtime / Real-time scan for Vulnerabilities and Misconfiguration and Remediate
- Secure Software Development
  - Whitelist/Blacklist Containers
  - Establish a secure container registry
  - Sign containers and code (MD5)
  - "Shift-left" vulnerability and bug scanning before deployment
- Log Management
  - Centralize container logs including developer actions
- Orchestration
  - Employ orchestration platform to manage containers across environments (DEV,TEST,QA,PROD) and across clouds

Container Security Solutions

- File System Compromise
  - Ensure file system is read only
  - Treat infrastructure as stateless, ideally serverless
- Networking
  - Ensure application containers can only talk to other approved application containers
  - Leverage Namespaces and SDN in orchestration tools
- Run Time Compromise / Privilege Escalation
  - Set filter on Linux Kernel to prevent privilege escalation and implement white lists
  - Anomaly detection based on a deviation from a known baseline to prevent remote code execution

Microservices Security
Challenges and Solutions

- Decomposition of Applications
  - Need to decompose applications into microservices correctly, so they only do one thing well, driving development of secure code
  - Monolithic code with 1,000 DLLs needs to be decomposed into 1,000 microservices which makes it more secure and maintainable
- Interface-driven development
  - Need to have well defined REST API's to ensure microservices talk consistently to each other
  - Authentication of API's should leverage OAuth and other secure protocols

Real World Examples

Docker CI/CD Pipeline Overview w/ GitLab

Development | Version Control & Continuous Integration | Continuous Deployment


```
version: '3.7'

services:
  web:
    image: custom-nginx2:latest
    ports:
      - '8000:8000'
```

Code Pull/Push

GitLab

CI/CD Pipeline
(.gitlab-ci.yml)

GitLab Runners
Test Scripts
Image Build
Image Sign/Push
Stack Deploy

RUNNER
Linux

RUNNER
Windows

Image Push

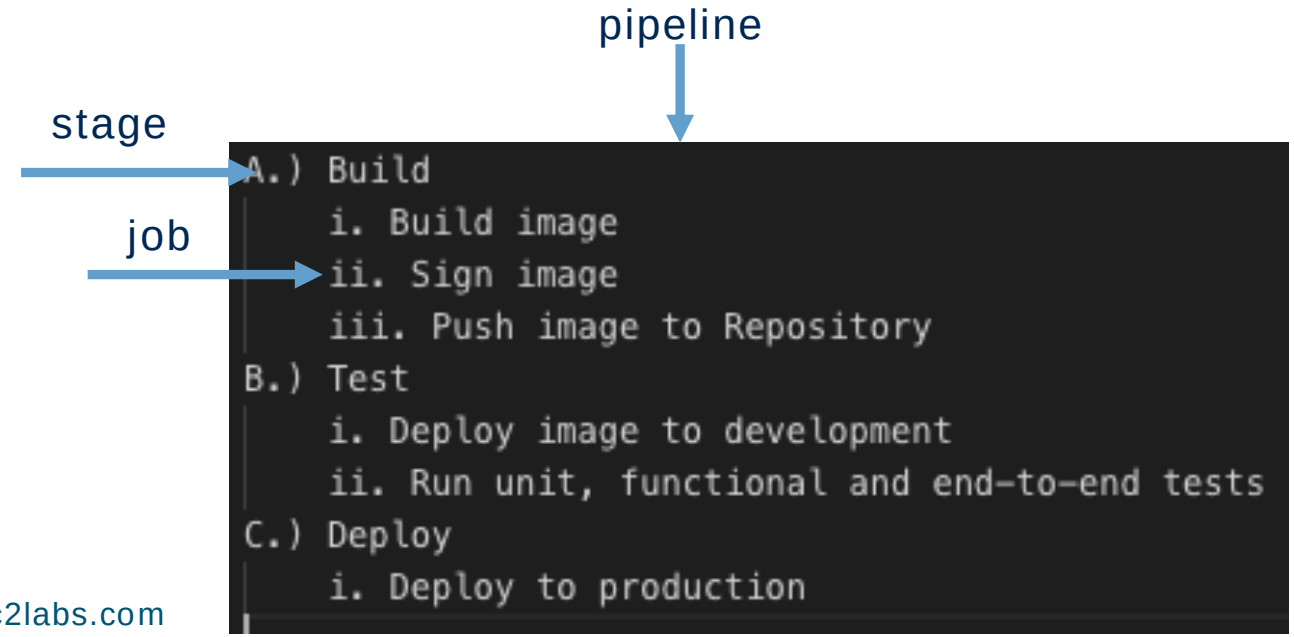
Deploy Stack

Docker Trusted Repository

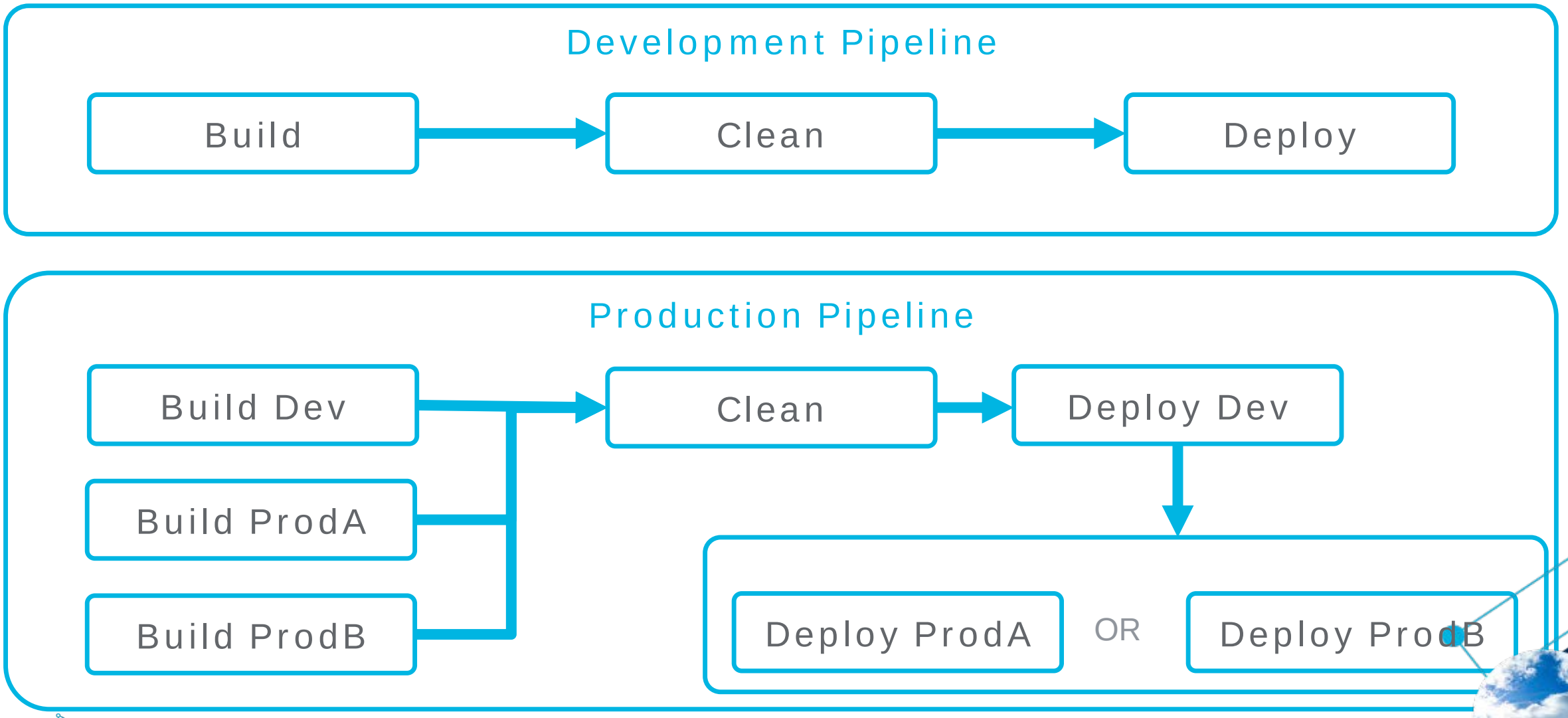
Docker Universal Control Plane

GitLab: What are Runners?

- Runners are the fundamental component of a CI/CD pipeline
- Runners are isolated virtual machines that run predefined steps through the GitLab CI API
  - Steps are defined in a .gitlab-ci.yml file
  - Steps execute as jobs, jobs are grouped together by stages, and stages are grouped together by pipelines
  - Job execution occurs on the Runner machine
  - Any dependencies/enablers that are required for a job to execute must be installed on the Runner machine

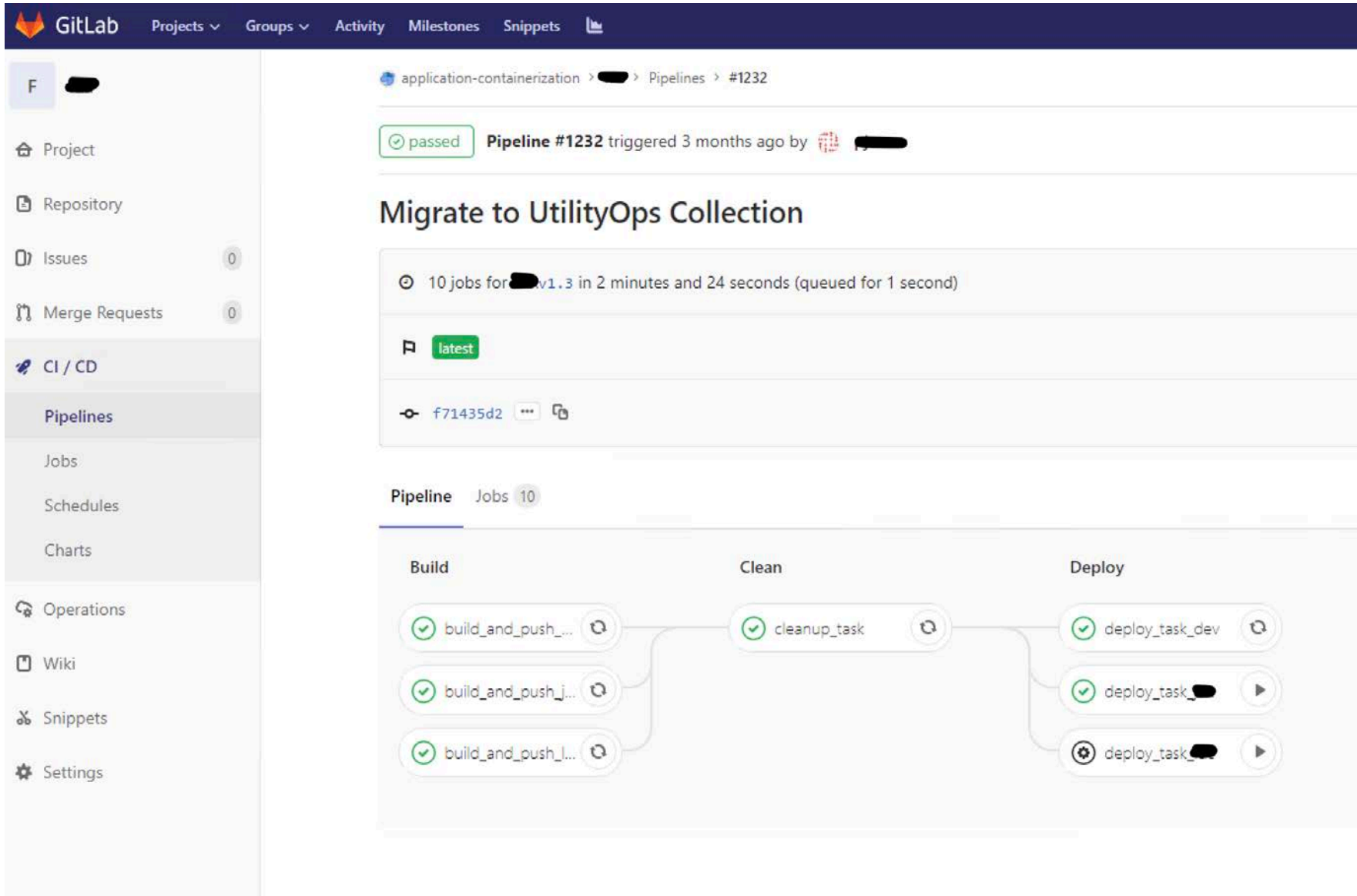


Docker CI/CD Pipeline Stages - Customizable



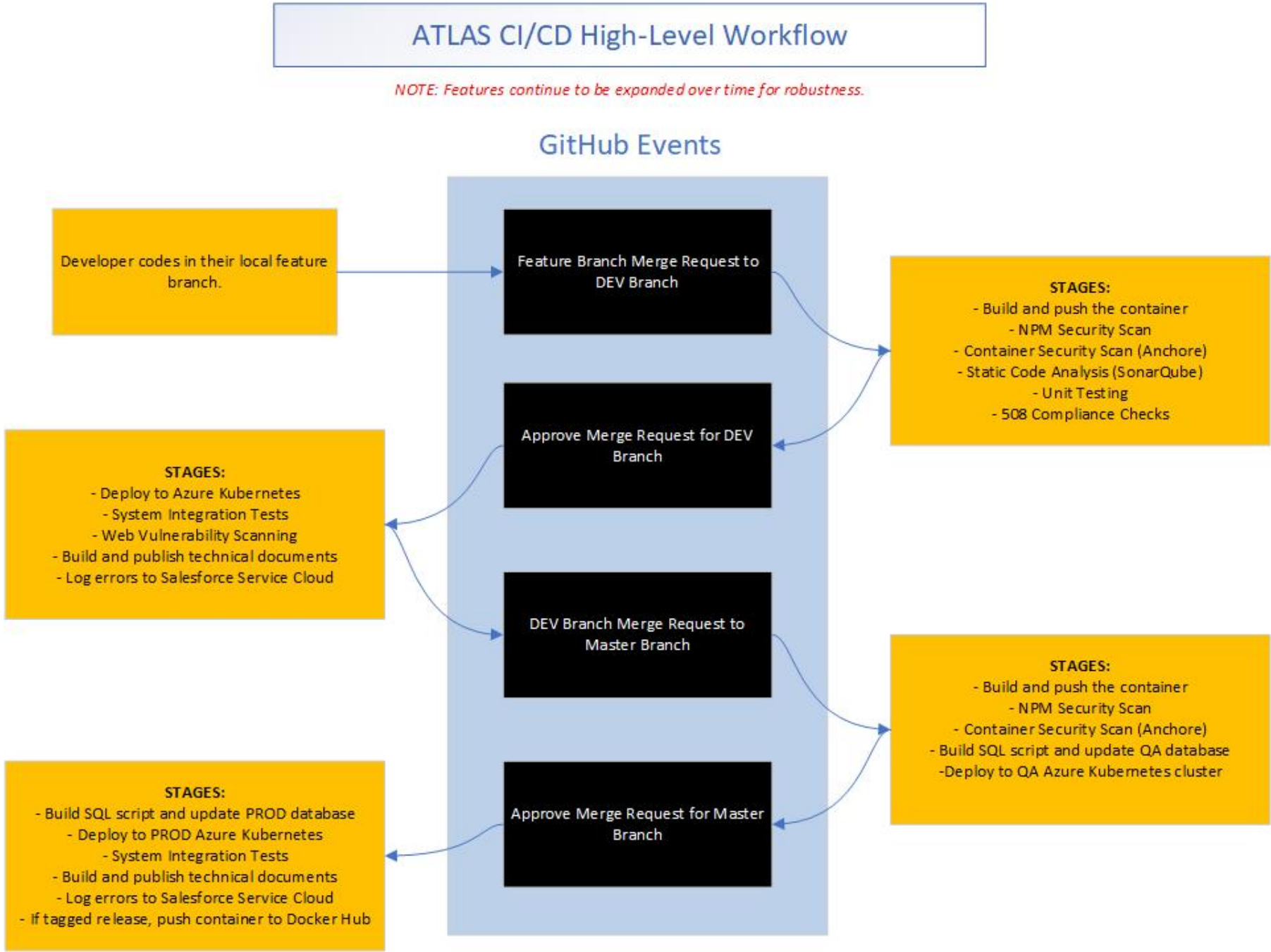
GitLab On-Premise CI/CD Pipeline

- Example of a production GitLab CI/CD pipeline
- All customizable; can implement your design easily
- (CI) Built and pushed image in DEV and both production environments
- (CI) Performed a clean build
- (CD) Deployed to DEV and the chosen production environment
- (CD) DEV deploy is triggered by commit to *dev* or *master* branch
- (CD) PROD deploy is triggered via a Tag and a Manual start by user with the right permissions



Commercial Tool Build Pipeline

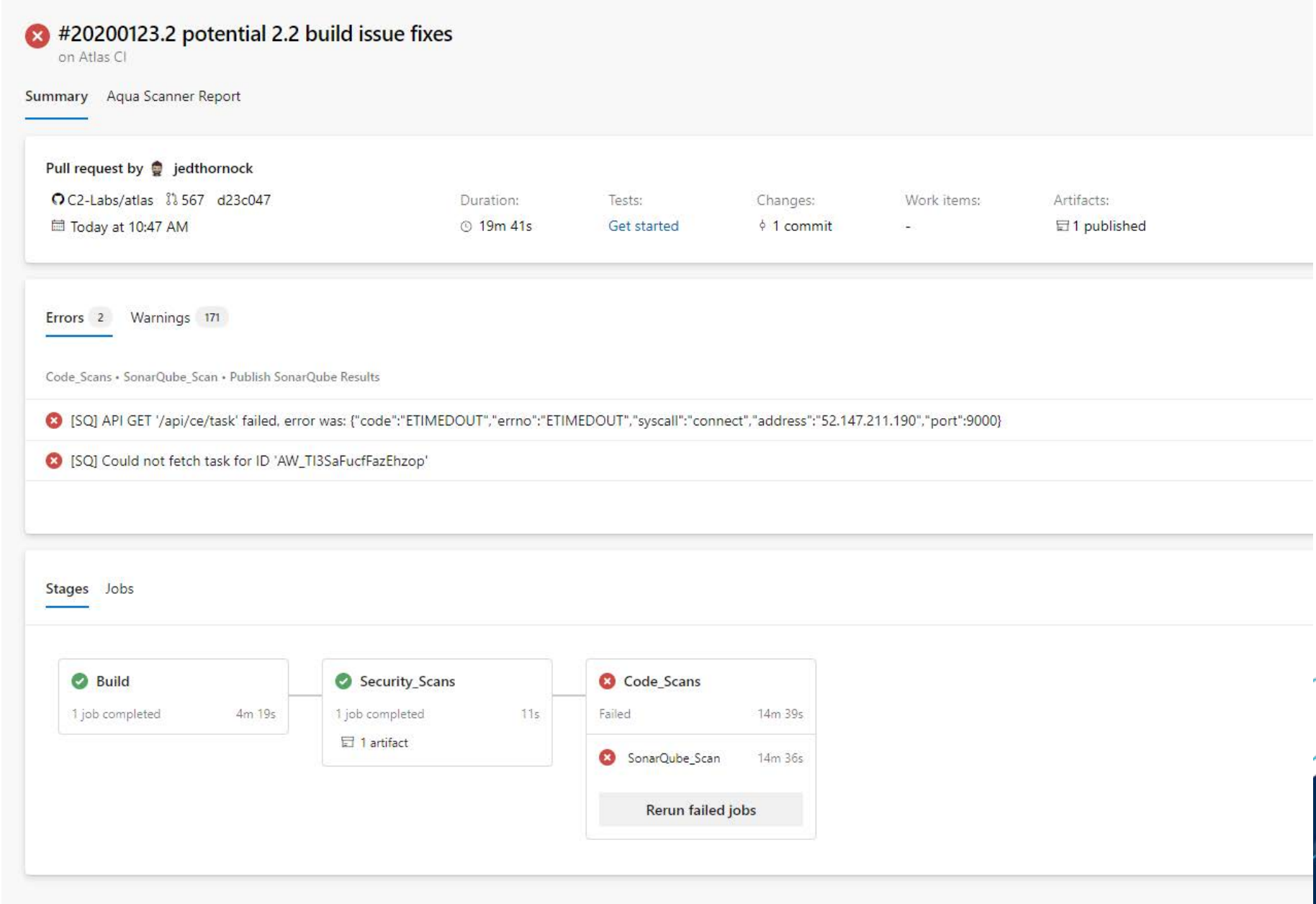
- CI/CD triggered based off of protected branch strategy
- CI steps generally deploy on a Pull Request (PR)
- CD steps generally deploy after approving the PR
- Testing, documentation, database upgrades, security scanning, logging, and Kubernetes deployment are done from the Feature branch -> DEV -> QA -> PROD with no manual labor; unlocking our developers full potential
- Governance is employed by adding workflow approvals to PRs
- All logs are maintained in Azure DevOps for Configuration Management



C2 LABS

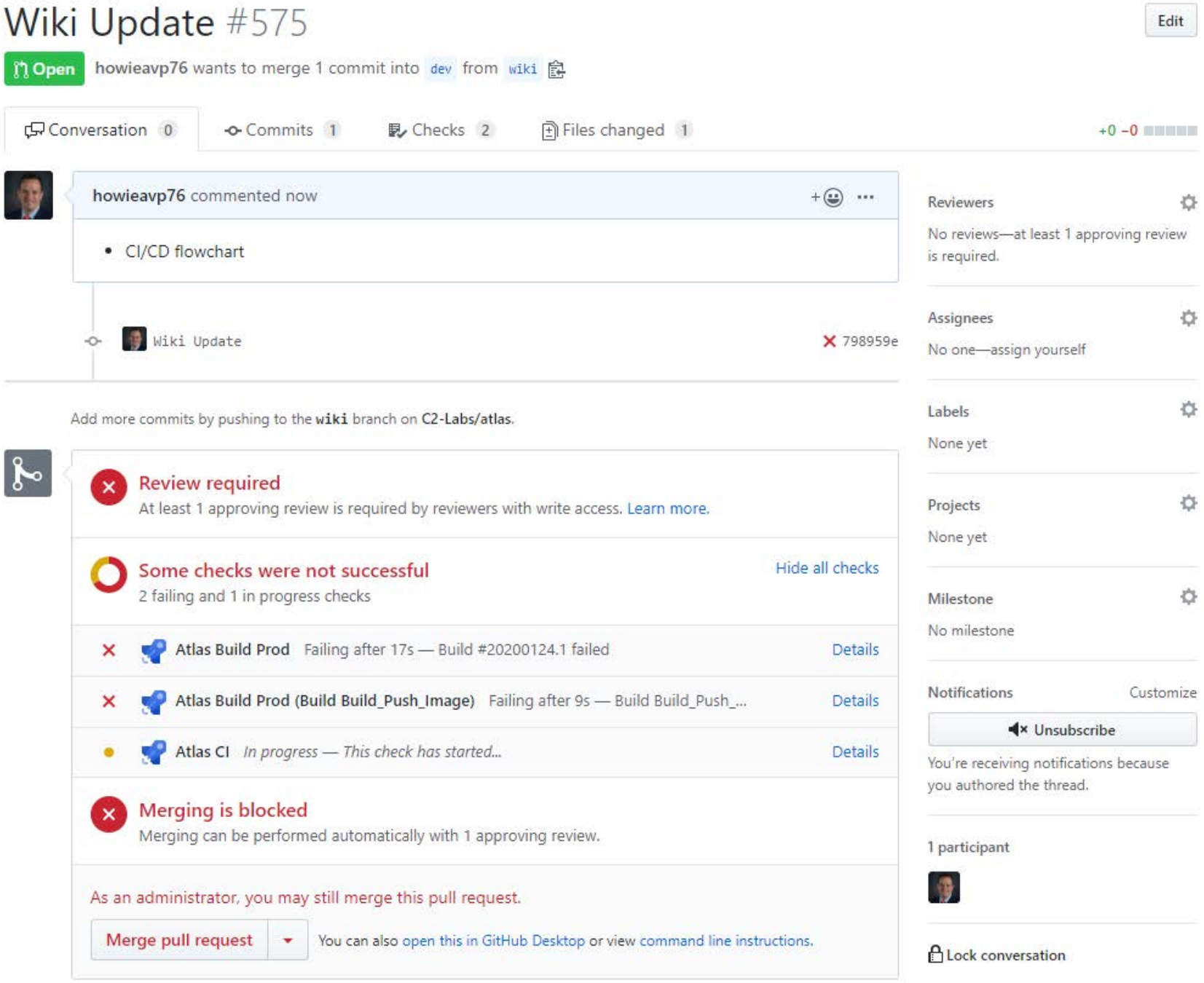
Multi-Stage Pipeline

- Container is built and pushed to our private Azure Registry
- Security scans are done via NPM Audit and Anchore
- Source code scan is done by SonarQube – pass/fail logic is coded into the stages
- Artifacts are stored for troubleshooting or later forensics if a defect escapes



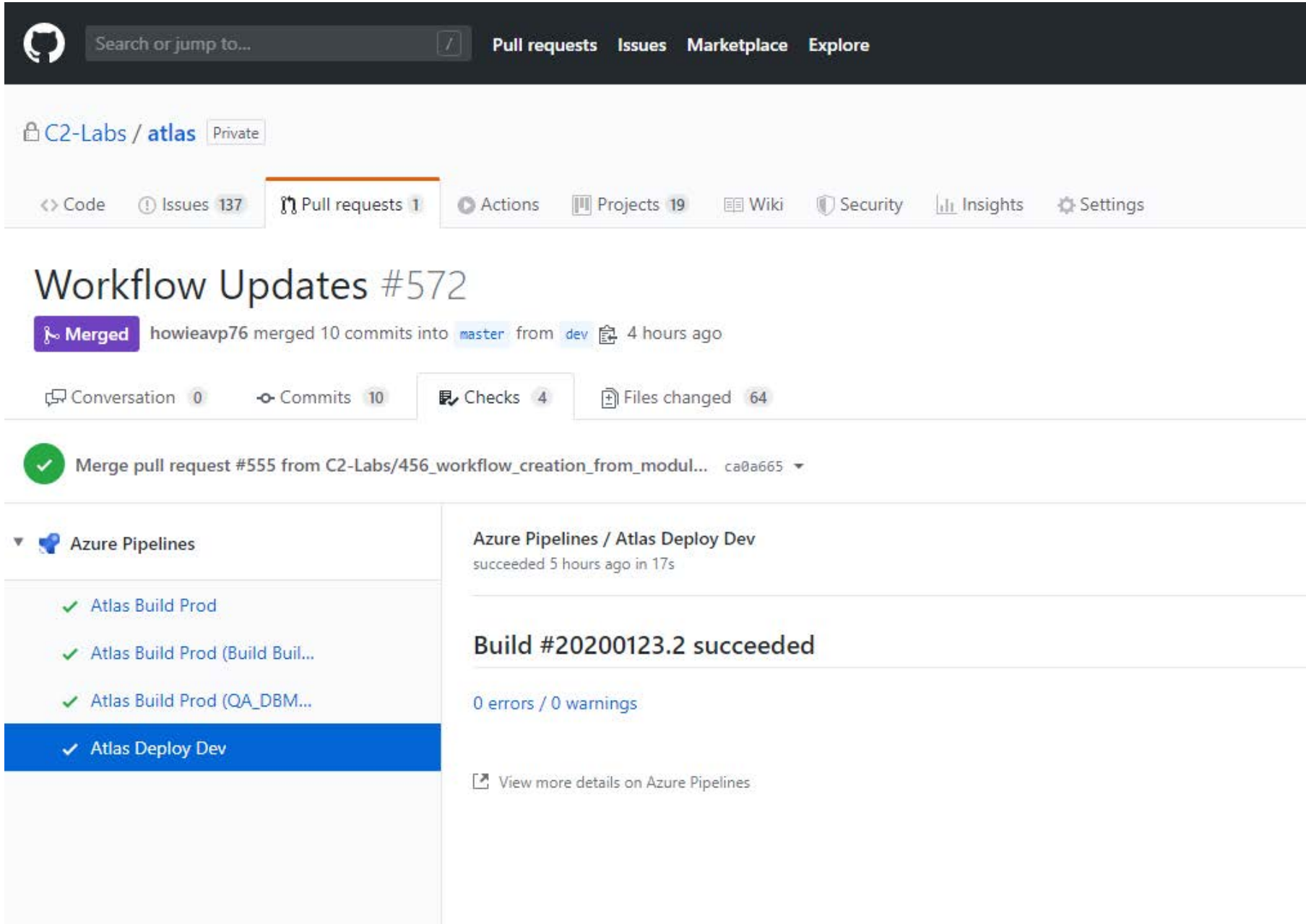
GitHub Integration

- Leveraged webhooks/plugins to show pipeline progress in the GitHub PR
- Details link to Azure DevOps to view artifacts and raw logs
- Governance enforces code reviews, pipeline checks passing, and two-person rule for a manager to approve code changes into a protected branch



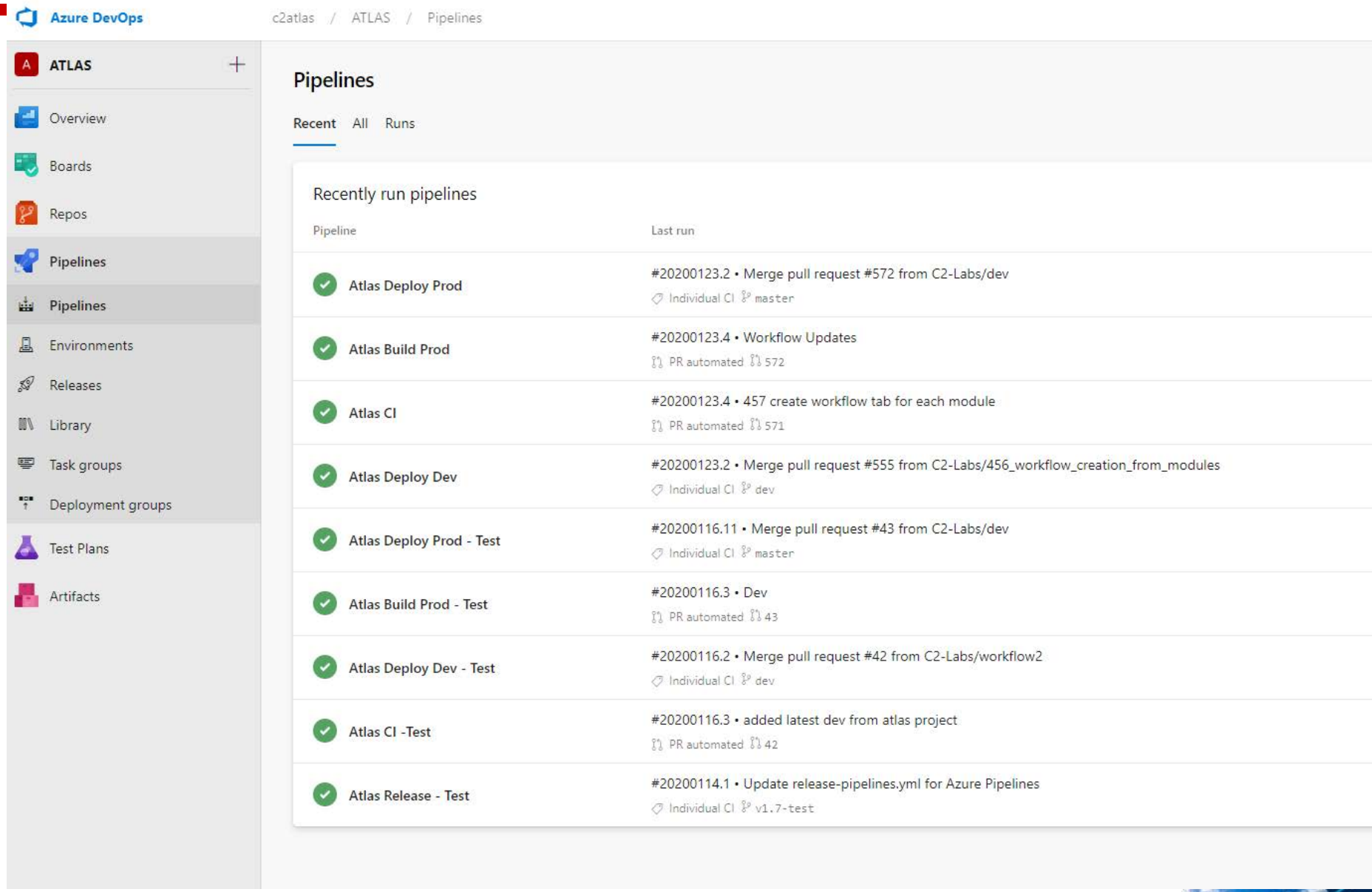
Integration with Azure DevOps

- Details pulled into GitHub and stored with the PR
- History maintained over time for full configuration management traceability
- Builds and deploys are tagged with the commit number to allow for easy rollbacks in Kubernetes



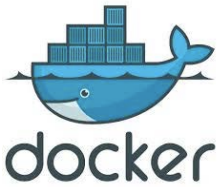
Azure DevOps Pipelines

- Multiple pipelines configured that are triggered based on GitHub branching logic
- Each pipeline has one or more stages to the job
- Each stage has one or more tasks that execute
- Pipeline configurations are developed in source code and under configuration management in GitHub
- NOTE: Pipeline changes are tested in a separate cloned project prior to being introduced into the Production pipeline



Definitions
Microservices and Containers

- Microservices
  - Decompose Complex Applications into Small, Independent Processes communicating with each other using language-agnostic API's
  - Highly Decoupled and Modular with services organized around capabilities (e.g. User Interface, Billing)
- Containers
  - Much like Virtualization abstracts the Operating System from Hardware, Containers abstracts Applications from the Operating System
  - Applications are isolated from other Applications on the same Operating System
  - Allows for Cloud Portability and Scale Up/Out
  - Security issues need to be evaluated and addressed in native container deployments



NIST and CSA Partnership

Researching Together to develop Best Practices

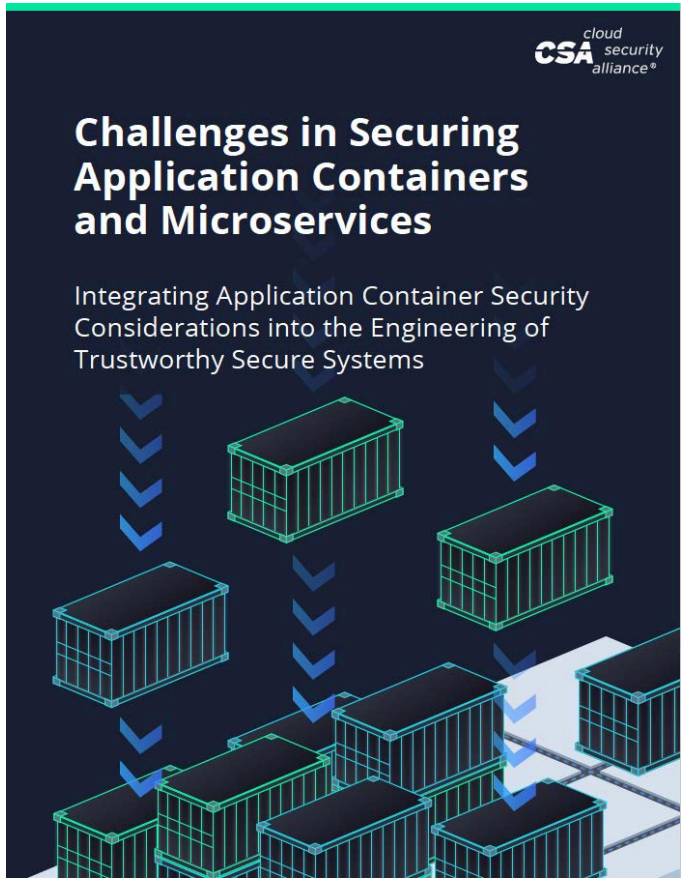
- NIST and CSA joined forces to define best practices for Application Containers and Microservices (ACM)
  - CSA ACM Members joined the NIST ACM Cloud Security Working Group
  - NIST artifacts served as the foundation for CSA ACM work
    - NIST SP 800-180: NIST Definition of Microservices, Application Containers and System Virtual Machines
    - NIST SP 800-190: Application Container Security Guide
    - NIST SP 800-160: Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Security Systems
    - NIST IR DRAFT: Challenges in Securing Application Containers and Microservices
    - NIST SP DRAFT: Best Practices in Securing Application Containers and Microservices

CSA Application Container and Microservices (ACM) Charter

CSA ACM Working Group Charter

- Objectives – Q1 2020
  - Best Practices to Implement a Secure Microservices Architecture
    - Microservices secure development guidance and governance
    - Best Practices for implementing a Microservices Architecture for Cloud-native applications
    - Best Practices for decomposing monolithic applications into Microservices

CSA Application Container and Microservices (ACM) Publications



Click on Titles to download the publications

Challenges in Securing Application Containers and Microservices

Best Practices for Implementing a Secure Application Container Architecture

Container Security Challenges

- Increased Attack Surface
  - Containers are far more complex than VM's wherein a single Application can consist of 1000's of microservices
  - Underlying Linux Operating System complexities can be exploited by attackers to compromise all containers on a host OS
  - Runtime Compromise / Vulnerabilities / Misconfiguration
- Secure Software Development
  - Containers can have code pushed to them from untrusted sources
- Log Management
  - Big Data Problem: How do you view and manage logs across 1000's of containers
- Orchestration
  - Infrastructure now runs as code (Puppet/Chef/Ansible)
  - Software developers, not infrastructure staff now run the data center

Container Security Challenges

- File System Compromise
  - Microservices in the Application Container could be compromised by an attacker
- Networking
  - A compromised container could result in lateral movement
- Run Time Compromise / Privilege Escalation
  - An attacker could modify a microservice in an Application Container which compromises the application or container itself

Container Security Solutions

- Increased Attack Surface
  - Employ MicroVM's (Just Enough VM)
  - Monitor Containers at Runtime / Real-time scan for Vulnerabilities and Misconfiguration and Remediate
- Secure Software Development
  - Whitelist/Blacklist Containers
  - Establish a secure container registry
  - Sign containers and code (MD5)
  - "Shift-left" vulnerability and bug scanning before deployment
- Log Management
  - Centralize container logs including developer actions
- Orchestration
  - Employ orchestration platform to manage containers across environments (DEV,TEST,QA,PROD) and across clouds

Container Security Solutions

- File System Compromise
  - Ensure file system is read only
  - Treat infrastructure as stateless, ideally serverless
- Networking
  - Ensure application containers can only talk to other approved application containers
  - Leverage Namespaces and SDN in orchestration tools
- Run Time Compromise / Privilege Escalation
  - Set filter on Linux Kernel to prevent privilege escalation and implement white lists
  - Anomaly detection based on a deviation from a known baseline to prevent remote code execution

Microservices Security
Challenges and Solutions

- Decomposition of Applications
  - Need to decompose applications into microservices correctly, so they only do one thing well, driving development of secure code
  - Monolithic code with 1,000 DLLs needs to be decomposed into 1,000 microservices which makes it more secure and maintainable
- Interface-driven development
  - Need to have well defined REST API's to ensure microservices talk consistently to each other
  - Authentication of API's should leverage OAuth and other secure protocols

Real World Examples

Docker CI/CD Pipeline Overview w/ GitLab

Development | Version Control & Continuous Integration | Continuous Deployment


```
version: '3.7'

services:
  web:
    image: custom-nginx2:latest
    ports:
      - '8000:8000'
```

Code Pull/Push

GitLab

CI/CD Pipeline
(.gitlab-ci.yml)

Docker Trusted
Repository

GitLab Runners

Test Scripts
Image Build
Image Sign/Push
Stack Deploy

RUNNER

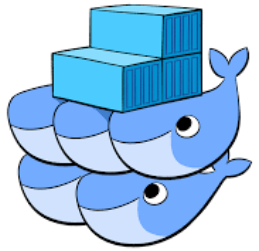
Linux

RUNNER

Windows

Image Push

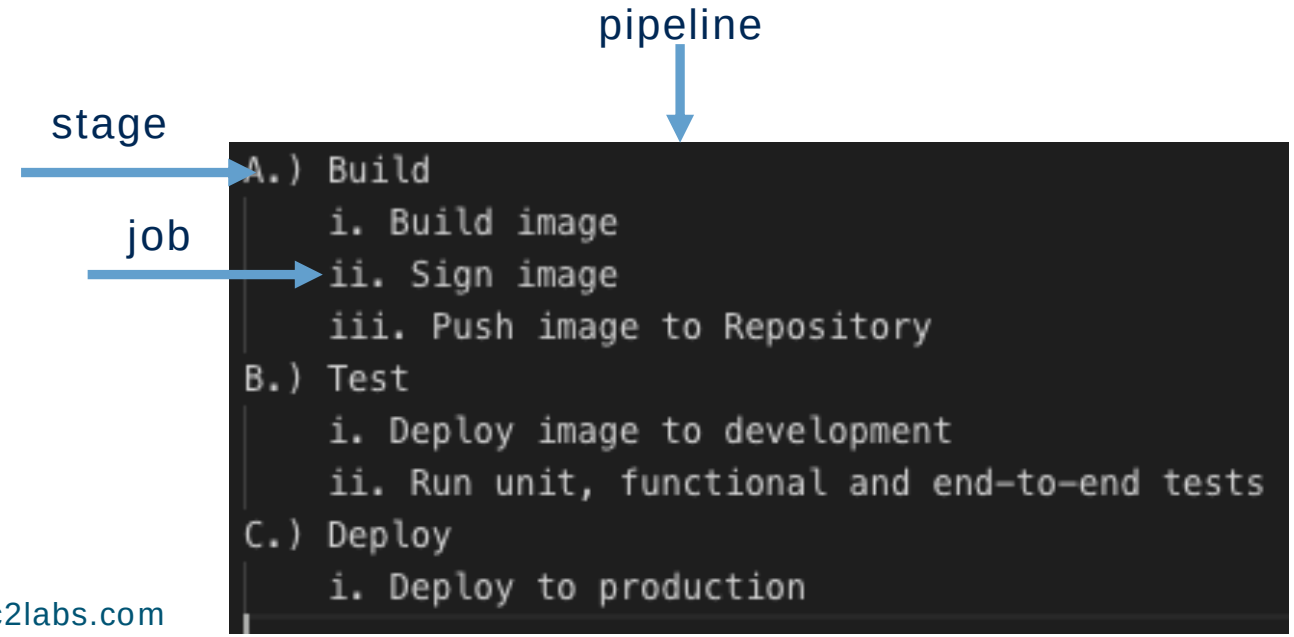
Deploy Stack



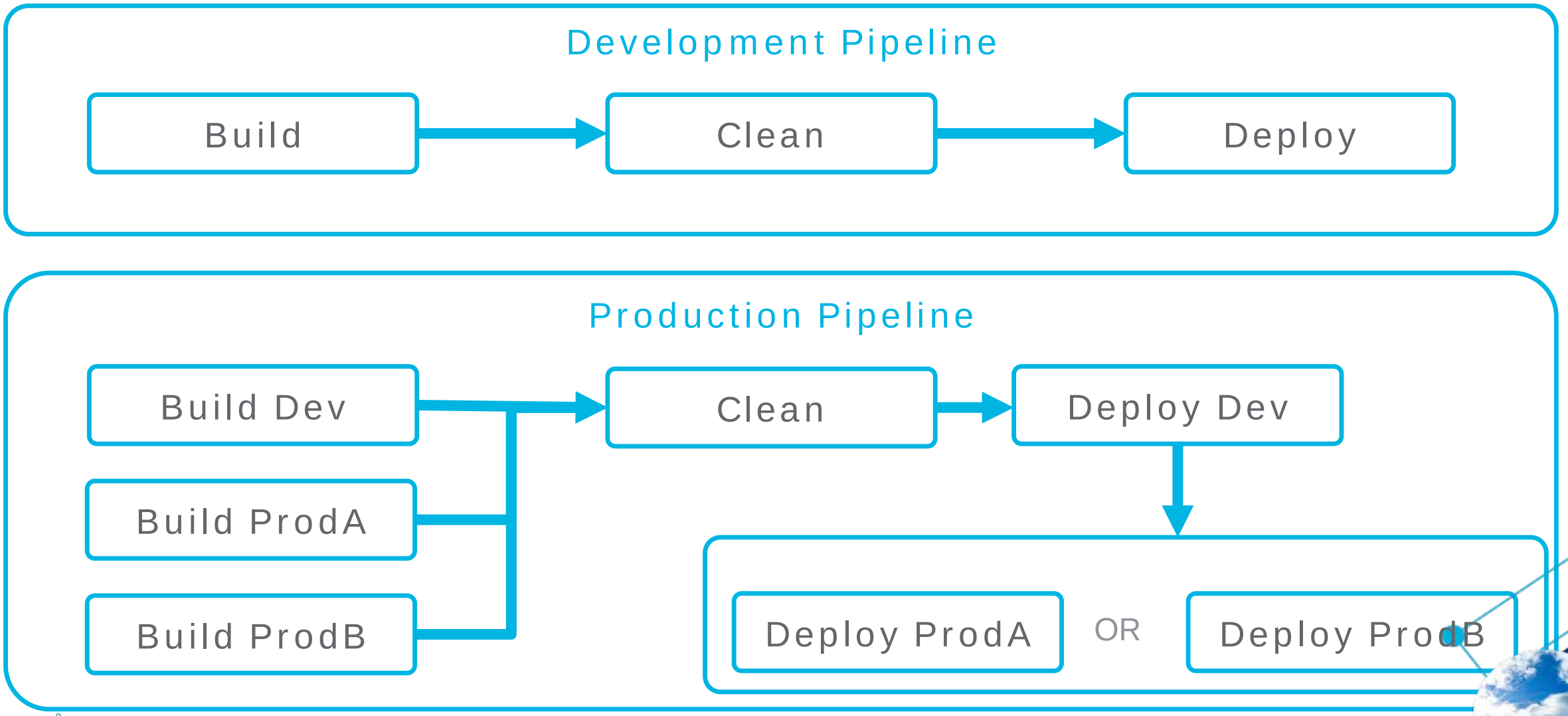
Docker Universal
Control Plane

GitLab: What are Runners?

- Runners are the fundamental component of a CI/CD pipeline
- Runners are isolated virtual machines that run predefined steps through the GitLab CI API
  - Steps are defined in a .gitlab-ci.yml file
  - Steps execute as jobs, jobs are grouped together by stages, and stages are grouped together by pipelines
  - Job execution occurs on the Runner machine
  - Any dependencies/enablers that are required for a job to execute must be installed on the Runner machine

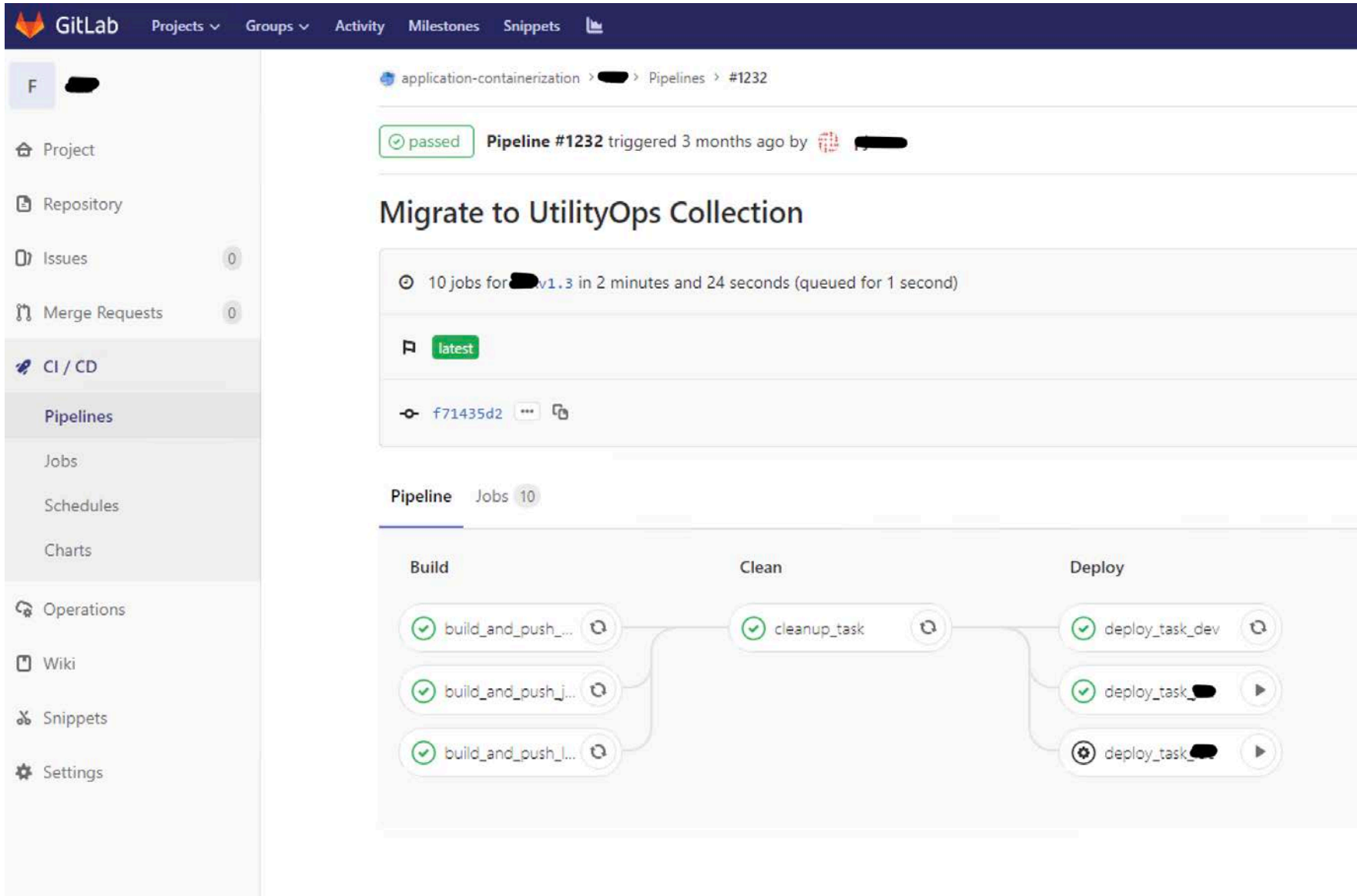


Docker CI/CD Pipeline Stages - Customizable



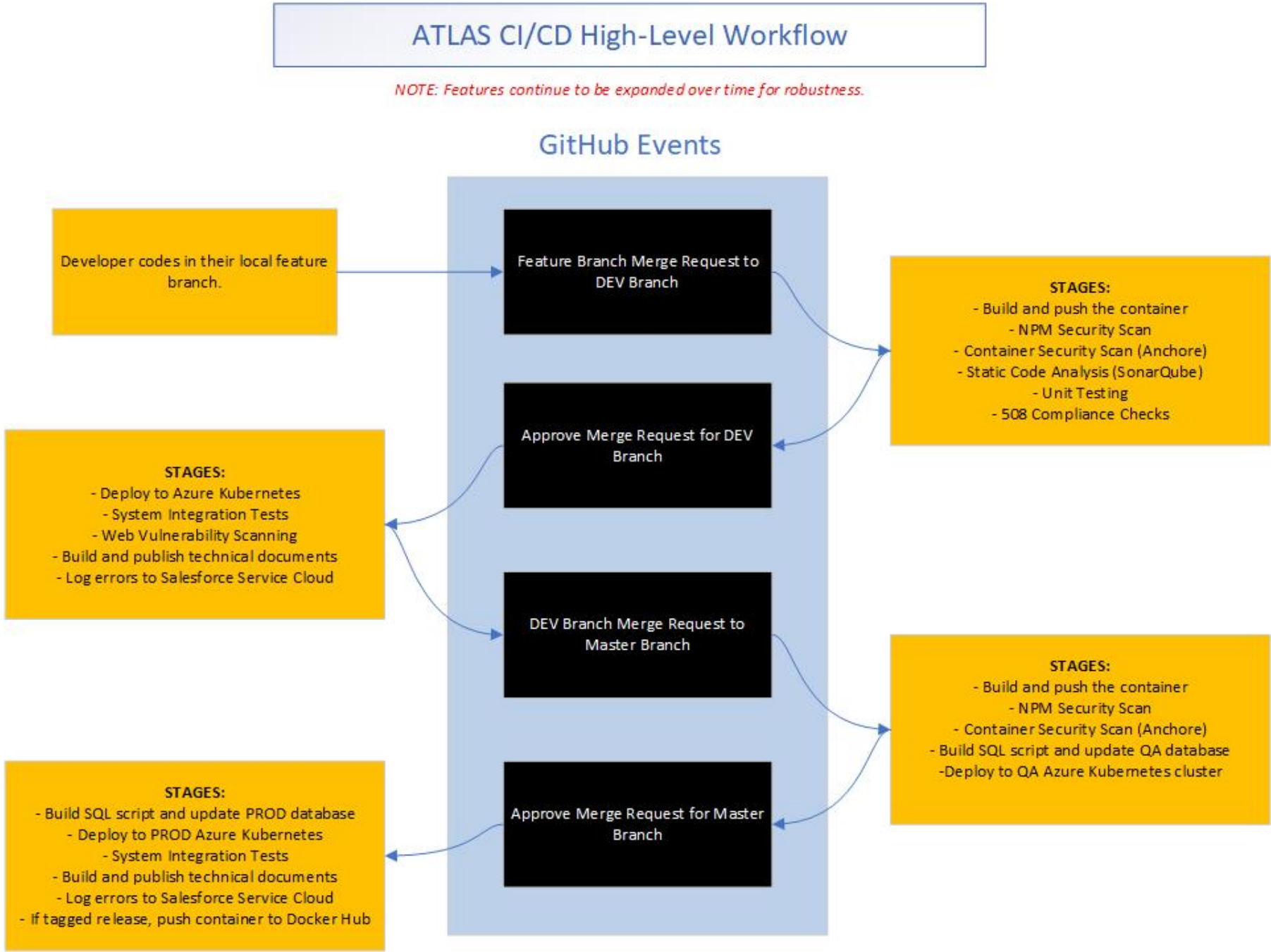
GitLab On-Premise CI/CD Pipeline

- Example of a production GitLab CI/CD pipeline
- All customizable; can implement your design easily
- (CI) Built and pushed image in DEV and both production environments
- (CI) Performed a clean build
- (CD) Deployed to DEV and the chosen production environment
- (CD) DEV deploy is triggered by commit to *dev* or *master* branch
- (CD) PROD deploy is triggered via a Tag and a Manual start by user with the right permissions



Commercial Tool Build Pipeline

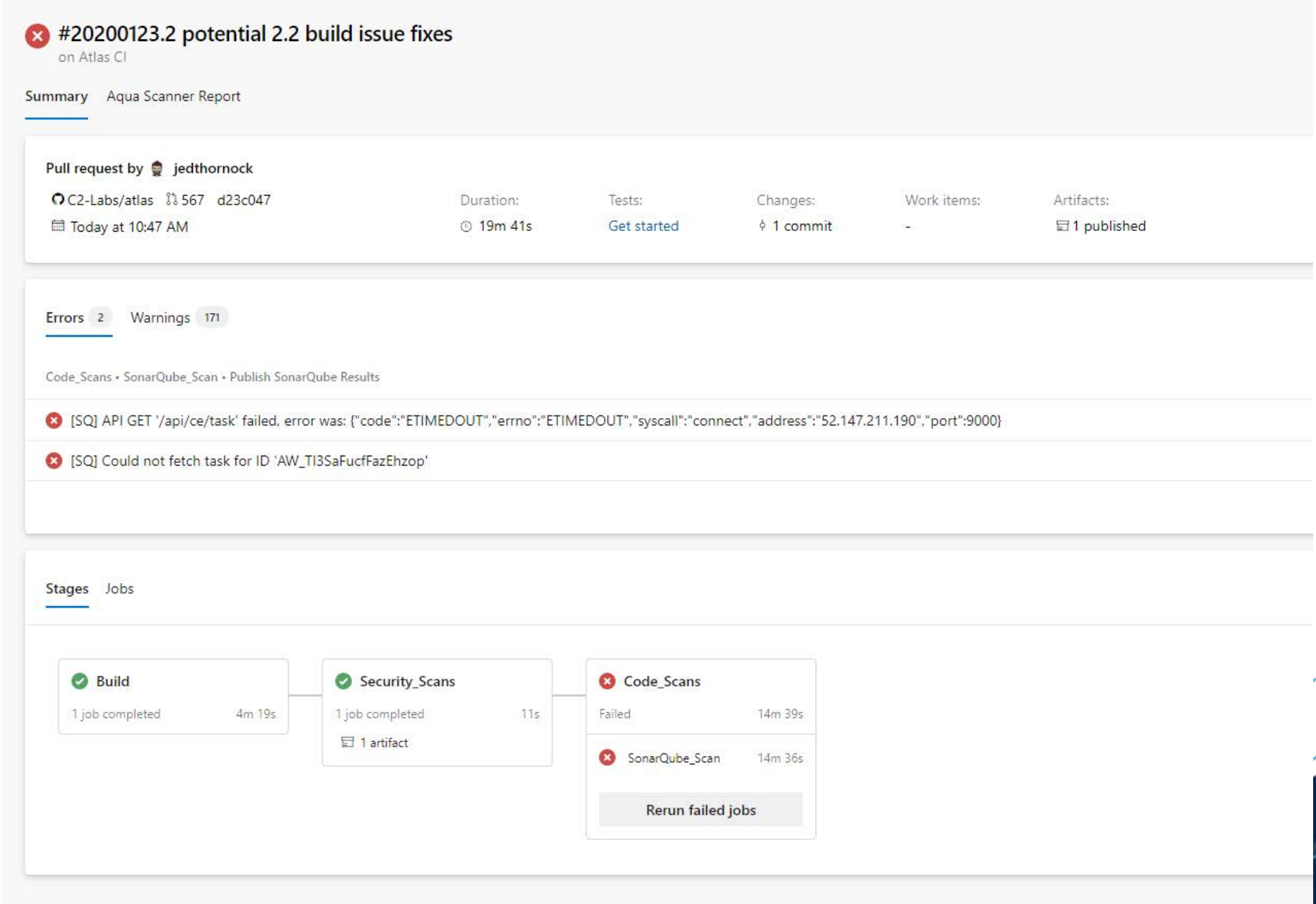
- CI/CD triggered based off of protected branch strategy
- CI steps generally deploy on a Pull Request (PR)
- CD steps generally deploy after approving the PR
- Testing, documentation, database upgrades, security scanning, logging, and Kubernetes deployment are done from the Feature branch -> DEV -> QA -> PROD with no manual labor; unlocking our developers full potential
- Governance is employed by adding workflow approvals to PRs
- All logs are maintained in Azure DevOps for Configuration Management



C2 LABS

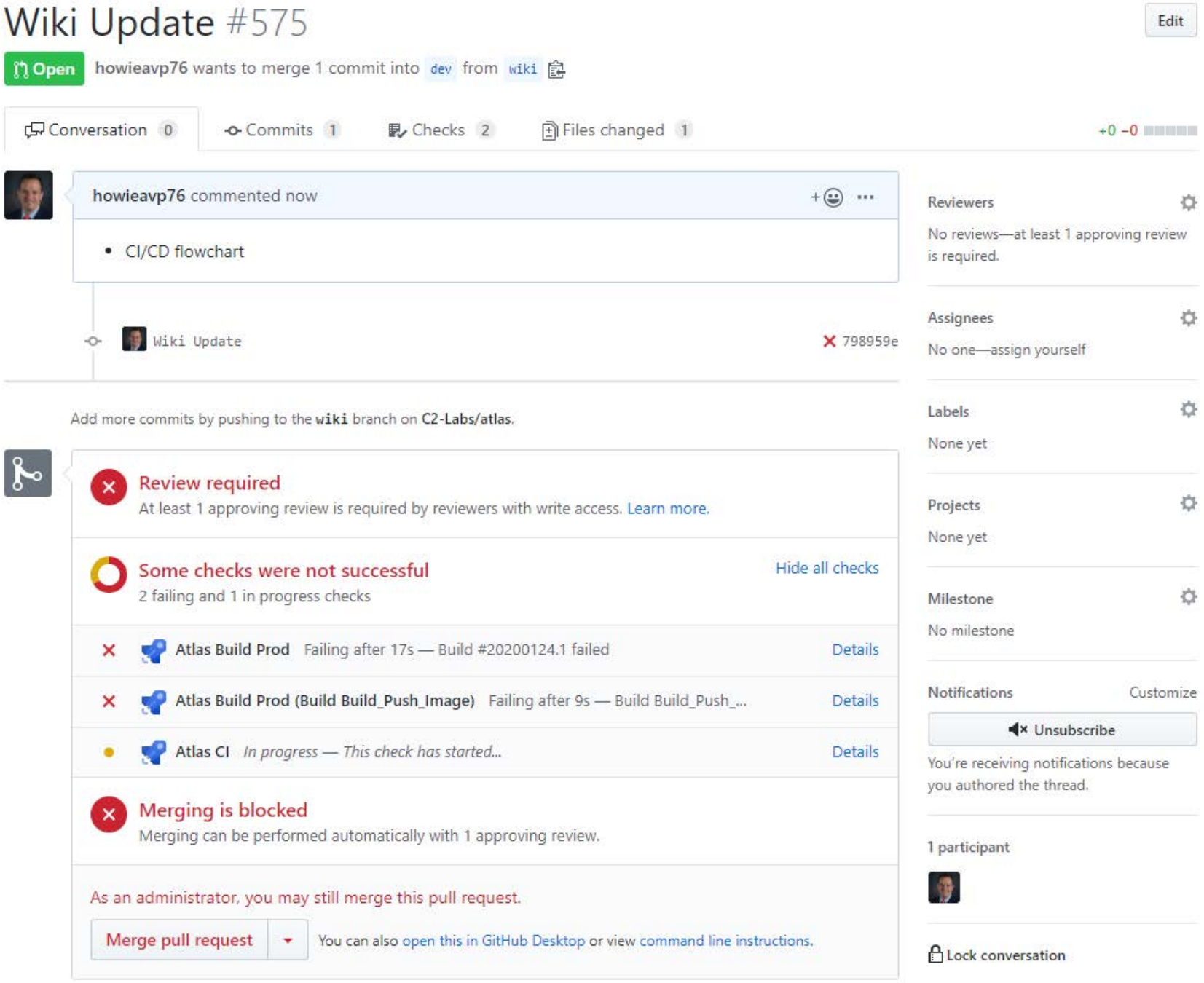
Multi-Stage Pipeline

- Container is built and pushed to our private Azure Registry
- Security scans are done via NPM Audit and Anchore
- Source code scan is done by SonarQube – pass/fail logic is coded into the stages
- Artifacts are stored for troubleshooting or later forensics if a defect escapes



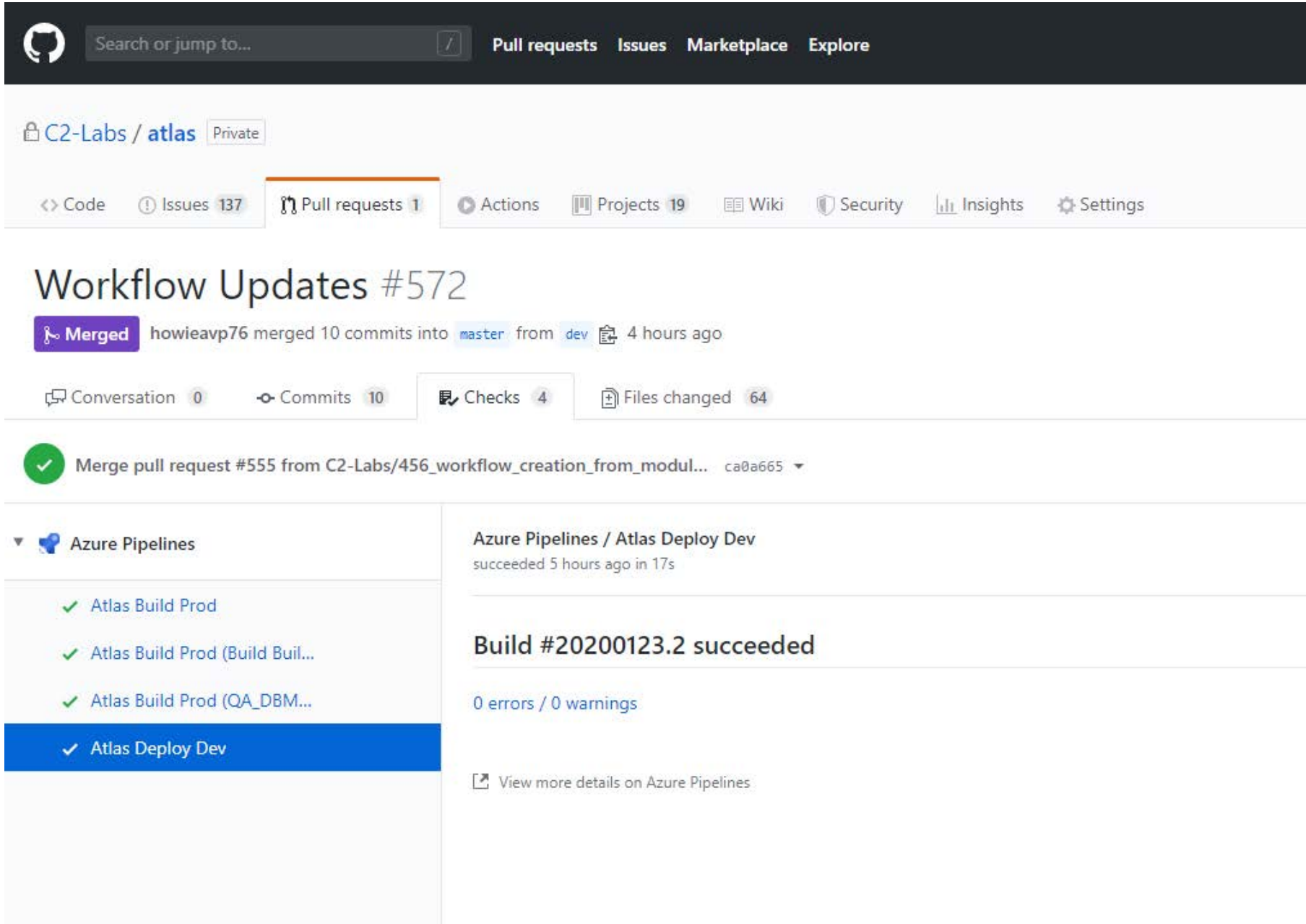
GitHub Integration

- Leveraged webhooks/plugins to show pipeline progress in the GitHub PR
- Details link to Azure DevOps to view artifacts and raw logs
- Governance enforces code reviews, pipeline checks passing, and two-person rule for a manager to approve code changes into a protected branch



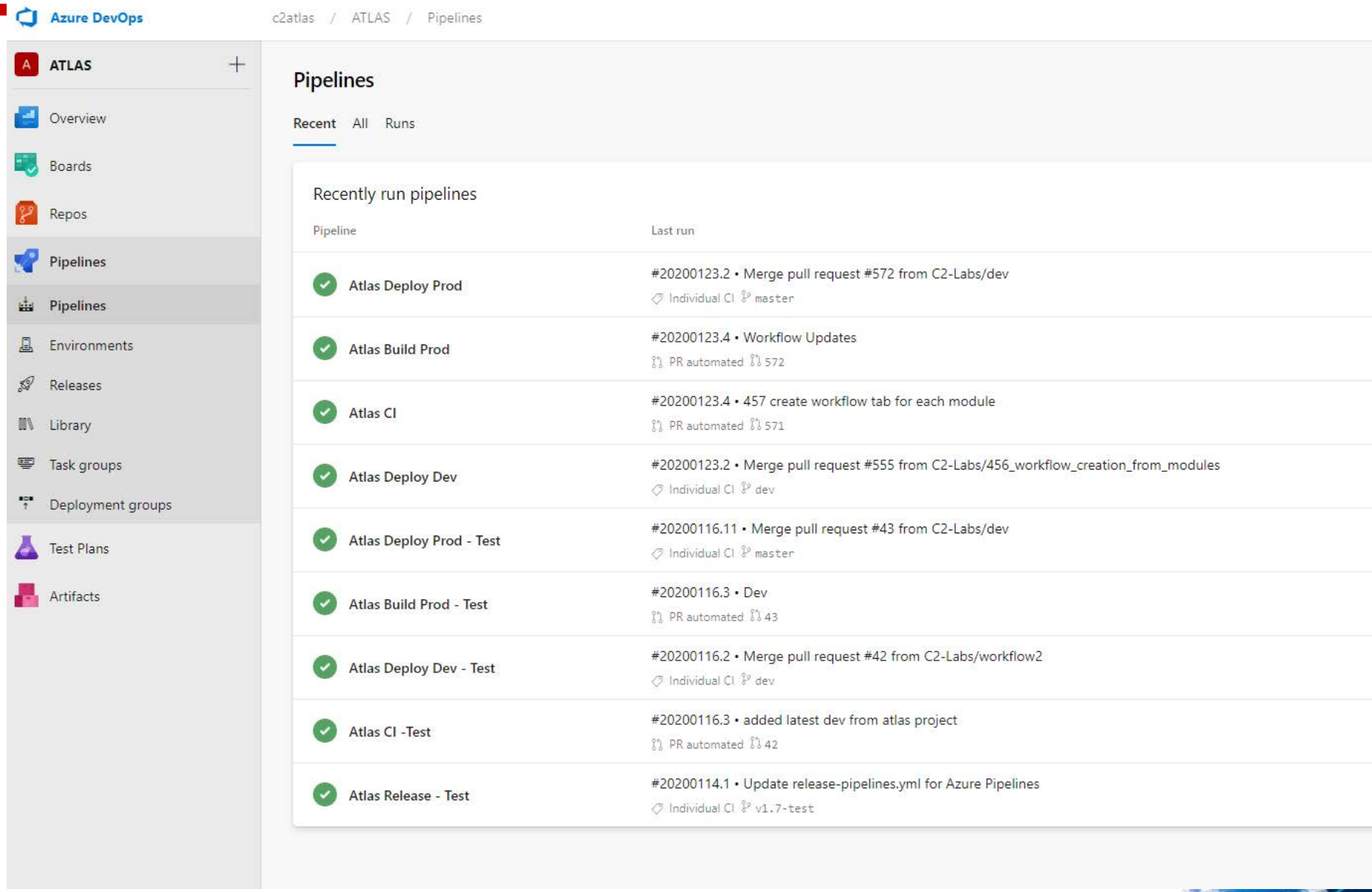
Integration with Azure DevOps

- Details pulled into GitHub and stored with the PR
- History maintained over time for full configuration management traceability
- Builds and deploys are tagged with the commit number to allow for easy rollbacks in Kubernetes



Azure DevOps Pipelines

- Multiple pipelines configured that are triggered based on GitHub branching logic
- Each pipeline has one or more stages to the job
- Each stage has one or more tasks that execute
- Pipeline configurations are developed in source code and under configuration management in GitHub
- NOTE: Pipeline changes are tested in a separate cloned project prior to being introduced into the Production pipeline



Stephen Naumann
Senior Advisor – Data Center Practitioner
GSA

Cloud Smart, Application Rationalization, and ICAM

National Institute of Standards and Technology
U.S. Department of Commerce

10:45 am

TETRATE

Office of Government-wide Policy

GSA OGP

DCOI, Cloud Smart, & ICAM

Steve Naumann, Senior Advisor | January 2020

Overview

Data Center and Cloud Optimization Initiative PMO

Data Center Optimization Initiative

Closure & Consolidation

Optimization

Cloud Smart

Workforce

Security

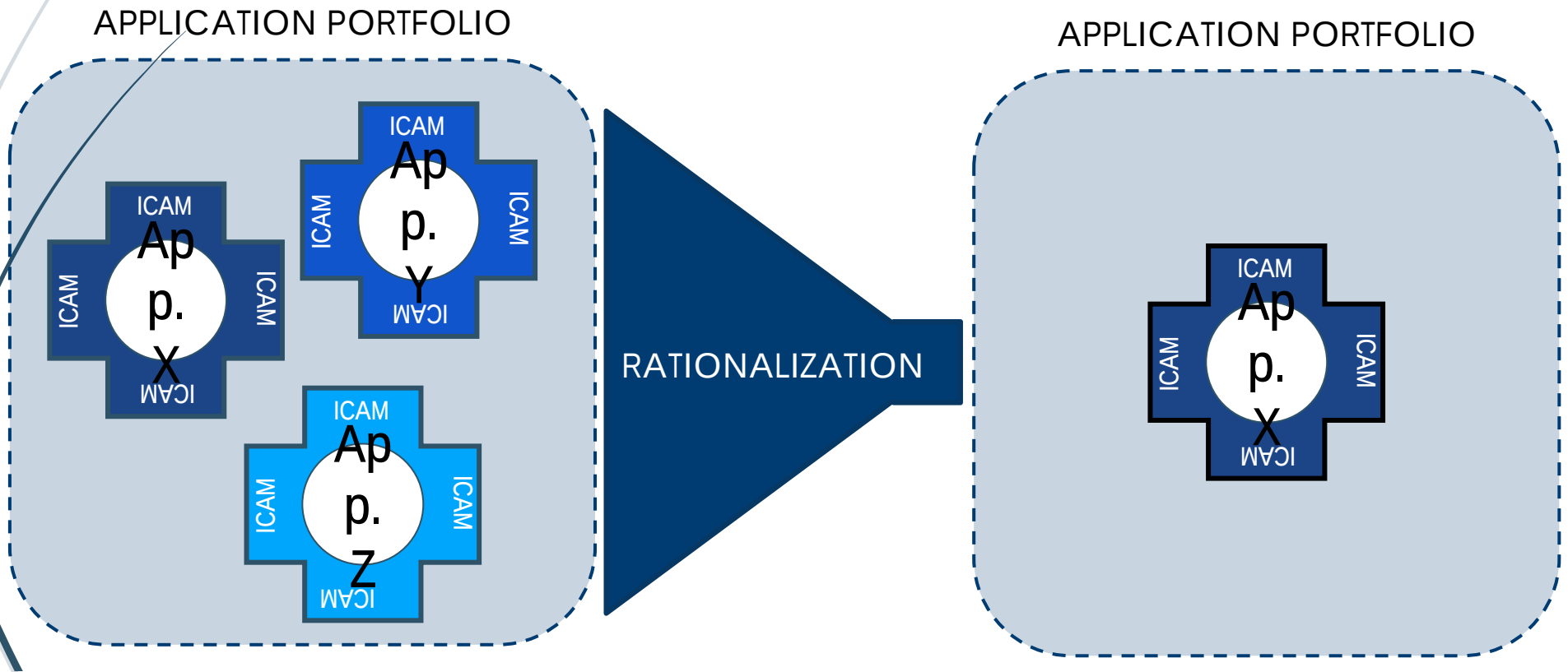
Procurement

Application Rationalization

Review; Reward; Refresh; Remove

Application Rationalization

- What is **Application Rationalization**?
- How is it connected to **ICAM**?



Cloud Policy Landscape

Implications for ICAM

Cloud Smart Strategy encourages:

- SLAs for access to log data
- Governance model aligned to ICAM systems

Other policies and paradigms:

- TIC 3.0
- Zero Trust Networks
- The Internet of Things

Cloud Adoption & ICAM

Challenges of Moving to Cloud

Recommended Process:

- Assess enterprise capabilities
  - Federation?
  - Fault tolerant?
  - Secure?
- Perform an ICAM gap analysis
- Address gaps (buy new or modify existing)

Goals in Common

Goal 1

Strengthen the Federal Government's information and physical security

1.1 Ensure that only authorized users can access protected resources

1.2 Enable agencies to establish and manage proven, trusted identities for all system users

1.3 Support the adoption and use of credentials that provide an efficient, secure means of accessing resources

1.4 Monitor user behavior and system security though diagnostics, analytics, and reporting

Goal 2

Enable information sharing and safeguarding within the Federal government and with external partners

2.1 Automate information discovery and access across the Federal Government in all security domains

2.2 Facilitate external partnerships by aligning ICAM business processes and technical interfaces with partners' best practices

2.3 Enable interoperability by standardizing information sharing agreements and establishing a common ICAM data architecture across government

Goal 3

Enable agencies to securely deliver mission services to customers

3.1 Design systems to allow customers frictionless access to information and resources

3.2 Foster trust by building protections for privacy and civil liberties into business processes and technical solutions

Goal 4

Support Federal Government efficiency in information technology

4.1 Streamline ICAM governance and program management within each agency or department

4.2 Standardize and automate ICAM business processes across the Federal Government

4.3 Establish shared service platforms and reuse or repurpose existing hardware and infrastructure when possible

Questions?
Email dccoi@gsa.gov

VIRTUAL AUDIENCE

Questions

(Slido)

National Institute of Standards and Technology
U.S. Department of Commerce

11:15 pm

TETRATE

Ad-Hoc Panel:
What is in Your Mind When You Think ZTA & DevSecOps

Moderator 1:
Jeyappragash Jeyakeerth
Co-Founder,
Tetrate

Moderator 2:
Michaela Iorga
Senior Security Technical Lead,
NIST

This is an ad-hoc panel and the panelists will be selected randomly from the members of the audience that expressed interest in participating in this dialog by registering in advance.

Panelist 1

Panelist2

Panelist3

Panelist4

11:30 am

Day 2
Closing Remarks and Adjourn

5:00 pm

National Institute of Standards and Technology
U.S. Department of Commerce

TETRATE