Journey of ECD

Jayachandran Natarajan

Journey of ECD

Apr, 2017

Jayachandran Natarajan

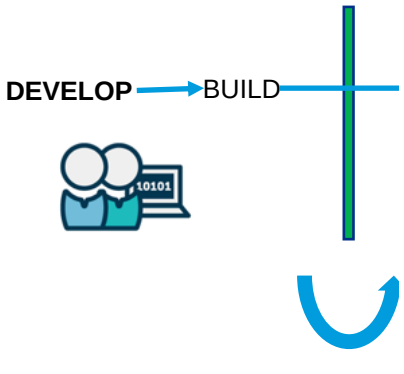
Agenda

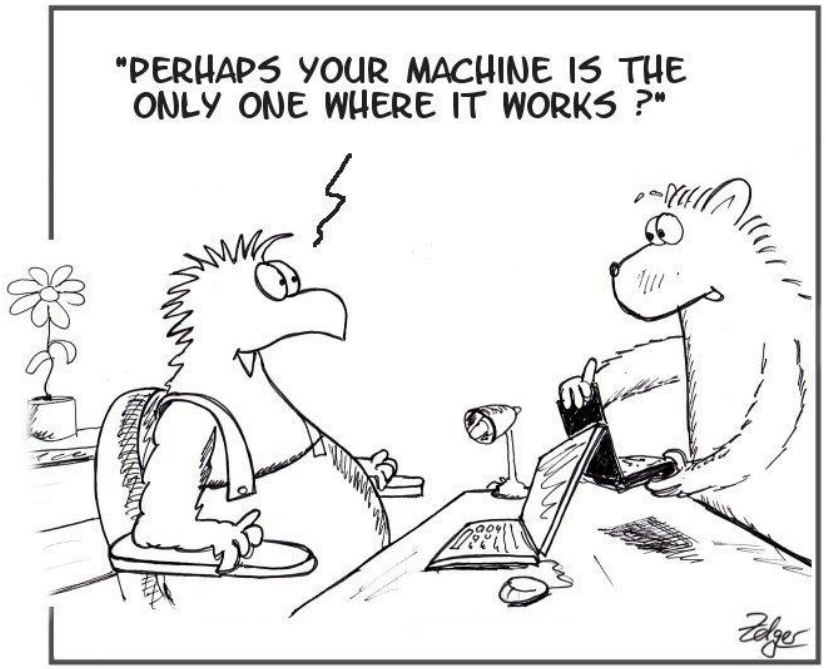
1. ECD?
2. Challenges in adoption
3. ALM Architecture
4. ECD
   - Features
   - Architecture
   - Pull Request & Post commit pipeline
   - Targeted CD Flow

ECD?

Continuous delivery (**CD**) is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time – *from Wikipedia*

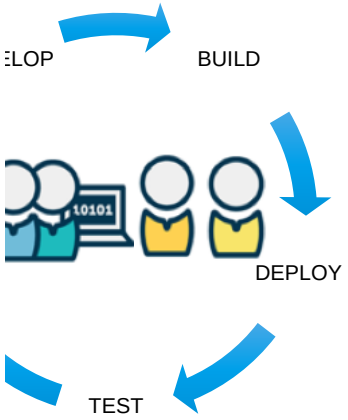
OL





It works on my machine

NEW WAY



http://simply-the-test.blogspot.in/2010/05/it-works-on-my-machine.html

CHALLENGES

Technical Challenges

- Monolith legacy builds
- Intricate deployments
- Single Jenkins master
- VM creation took ages
- Resource scaling was not agile

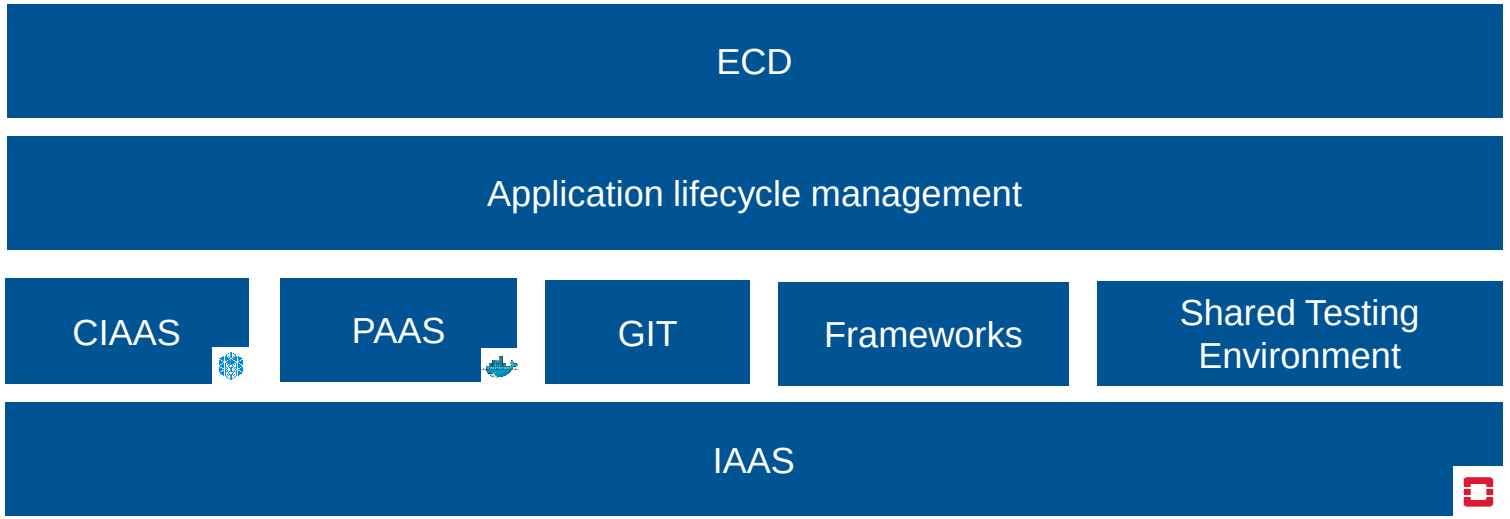
Process Challenges

- Application management complexity
- Lengthy release cycles
- Difficulty in audition
- Steep learning curve

PDLC Comparison

	2013	Now
Time to setup a new application *(including assets)*	~1-2 months	< 15 minutes
Build Time	~ hours	~ minutes
VM provisioning	Weeks to months	~ minutes
Release duration	~5-6 days	< 10 minutes
Release Cycle	Bi-Monthly	Anytime
Teams involved for release	Release Management, Release Engineering & Dev Teams	Any individual (on a single click)
SPOF	Yes	No
Feedback time on quality analysis	Minimum 1 day	< 30 minutes

Current Engineering Architecture



Infrastructure As A Service

- World largest private OpenStack cloud
- 4,00,000 cores
- 83,000 VMs
- Scalable

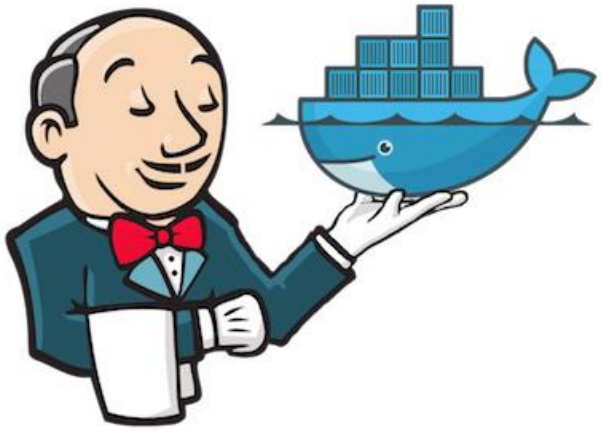
Shared Testing Environment

- Continuously available test environment with multitenant test nodes which will be constantly refreshed in sync with production

Frameworks

- Updated to support Isolated build & deploy model using bi-directional proxy

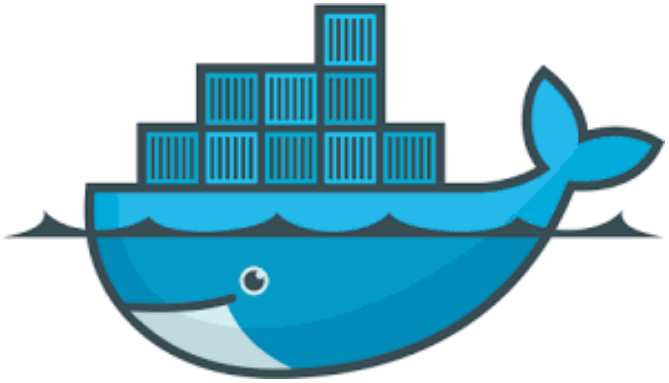
Continuous Integration As A Service



- 1:1 Jenkins Model
- Freedom to users
- Isolated build model
- Binary consistency
- Efficient resource utilization
- Self healing architecture

Platform As A Service

- Self service VMs in cloud
- Container based deployment
- Simplified APIs
- Efficient utilization of resources
- Instrumentation support



Application Life cycle management

- Deep Integration into PDLC
- Polyglot frameworks
- Create & Manage assets in single window
- Monitoring & Ownerships
- Scalable infrastructure

ECD - Why not Open Source tools?

In house CD tool using Spring Batch, Jersey & AngularJS with existing capabilities of Jenkins

Go

Pros

- Simple to use
- Valuable information at each stage
- Excellent GUI (good choice if GUI capabilities are leveraged)
- Easy to configure

Cons

- REST API documentation is not detailed enough
- Very few examples or tutorials available online
- Recently declared as Open source
- WIP for some APIs as seen in documentation

Jenkins

Pros

- Heavy customization possible
- Enormous plugins support available
- Very good Open source option (Online discussions available)

Cons

- Semantic is mainly CI oriented, must be heavily customized for what we need as CD
- Jobs needs to be manually configured for every type on steps

Bamboo

Pros

- Simple to use
- Nice GUI, which provides current job status, estimated time left, logs, etc…

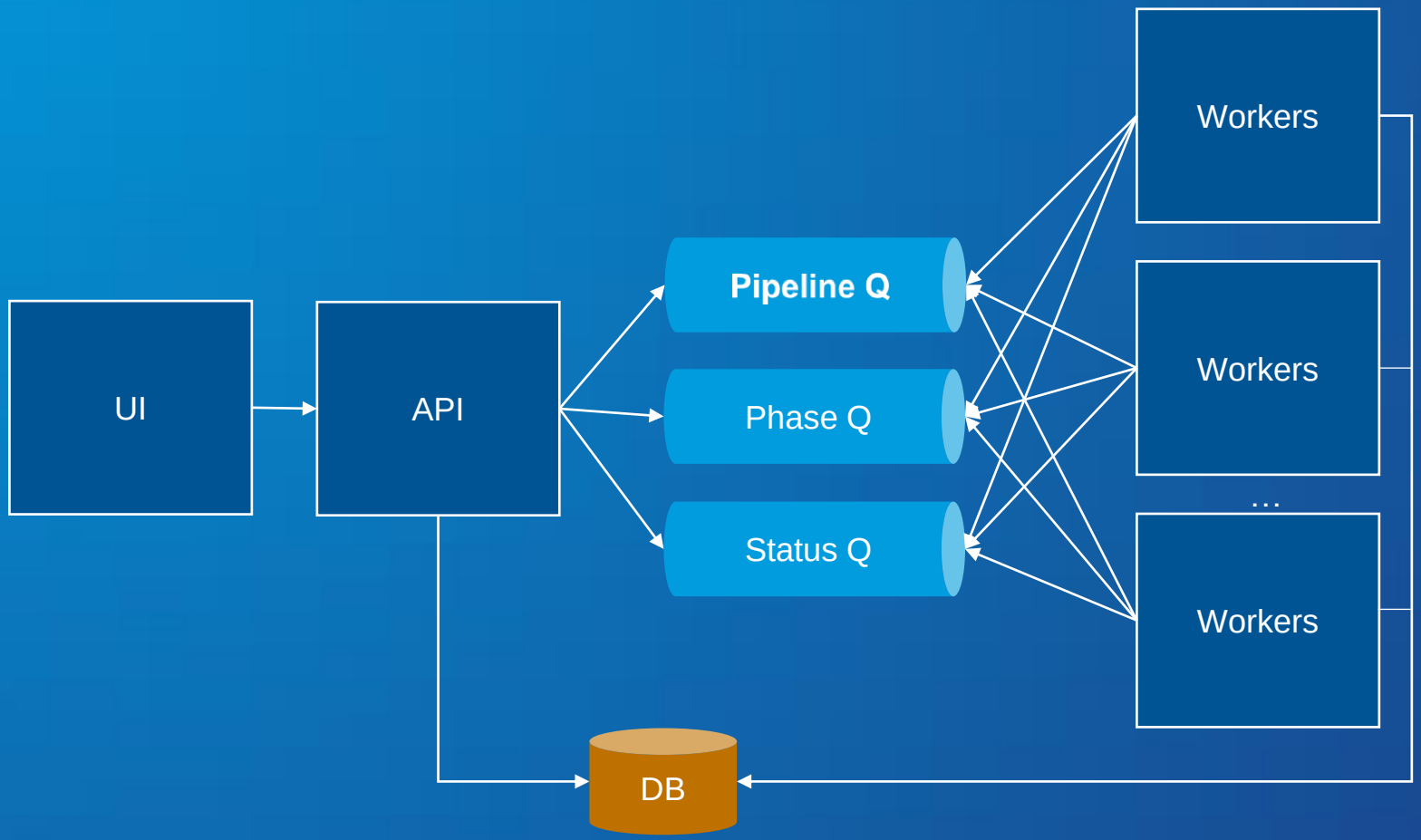
Cons

- Licensed
- Not sure about the customization feasibility

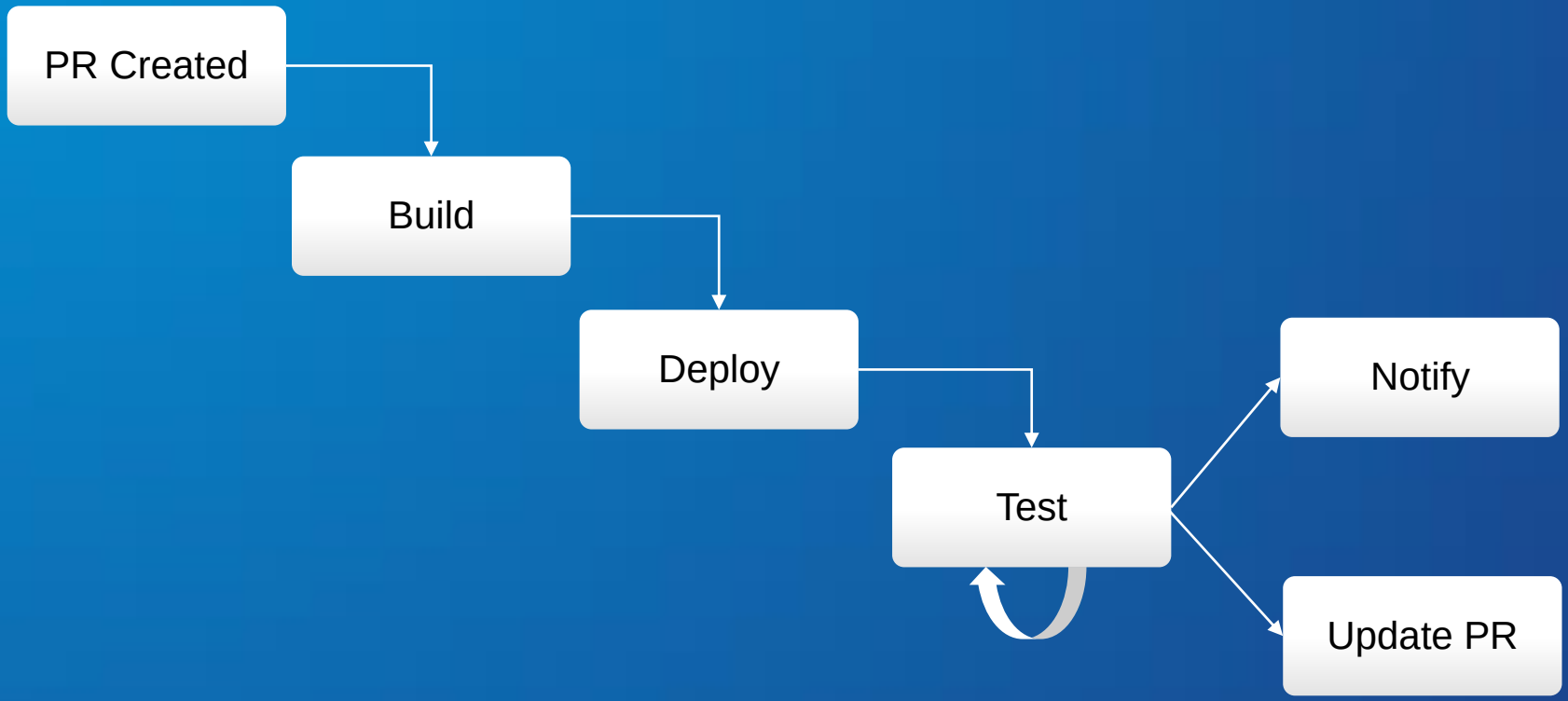
ECD - Features

- Automated creation flow
- Flexibility to extend steps
- Parallel processing
- Customizable verification points
- Shine the lights & links to actual task logs
- Alerts & Notifications
- Simple & Intuitive UI
- YAML based definition

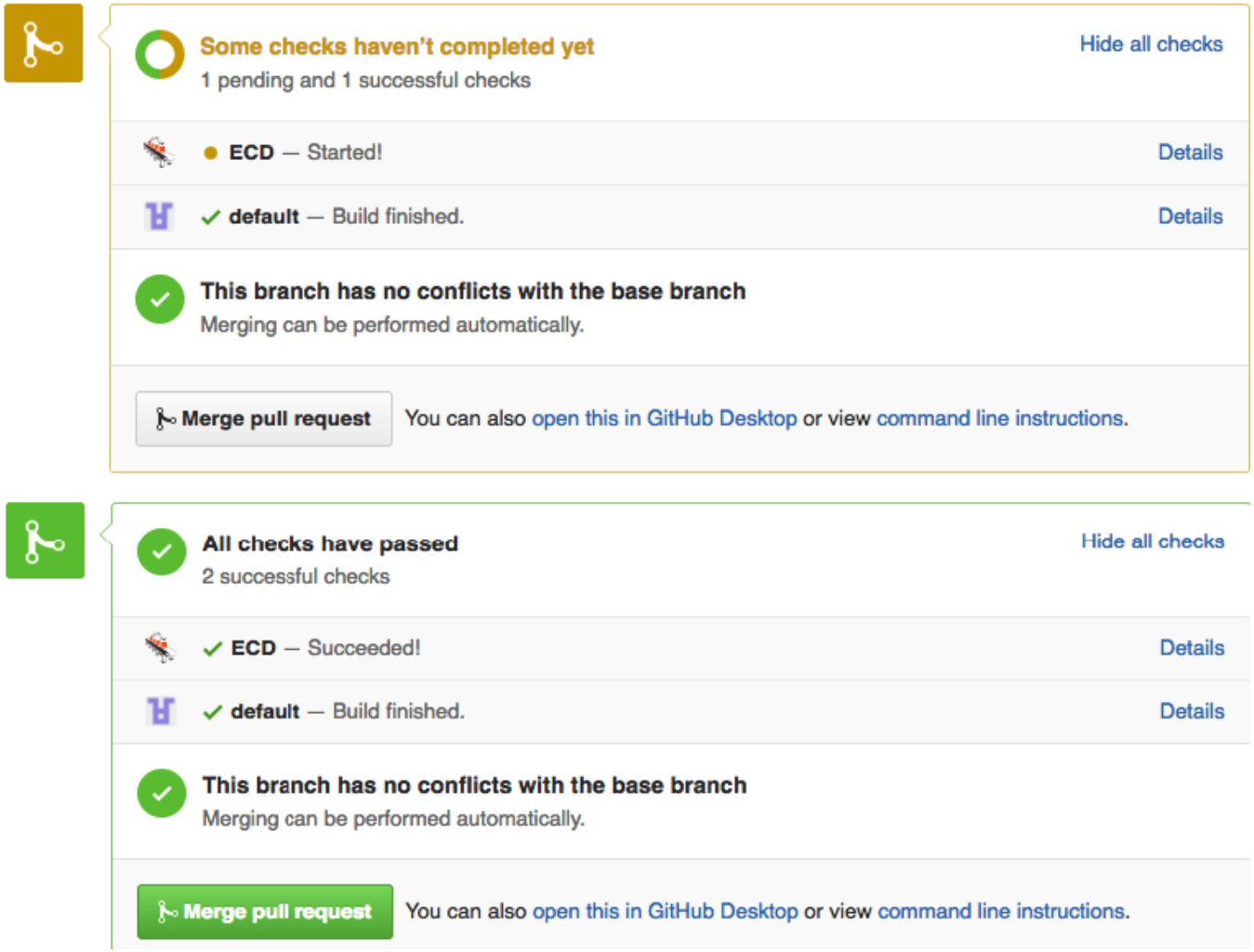
ARCHITECTURE



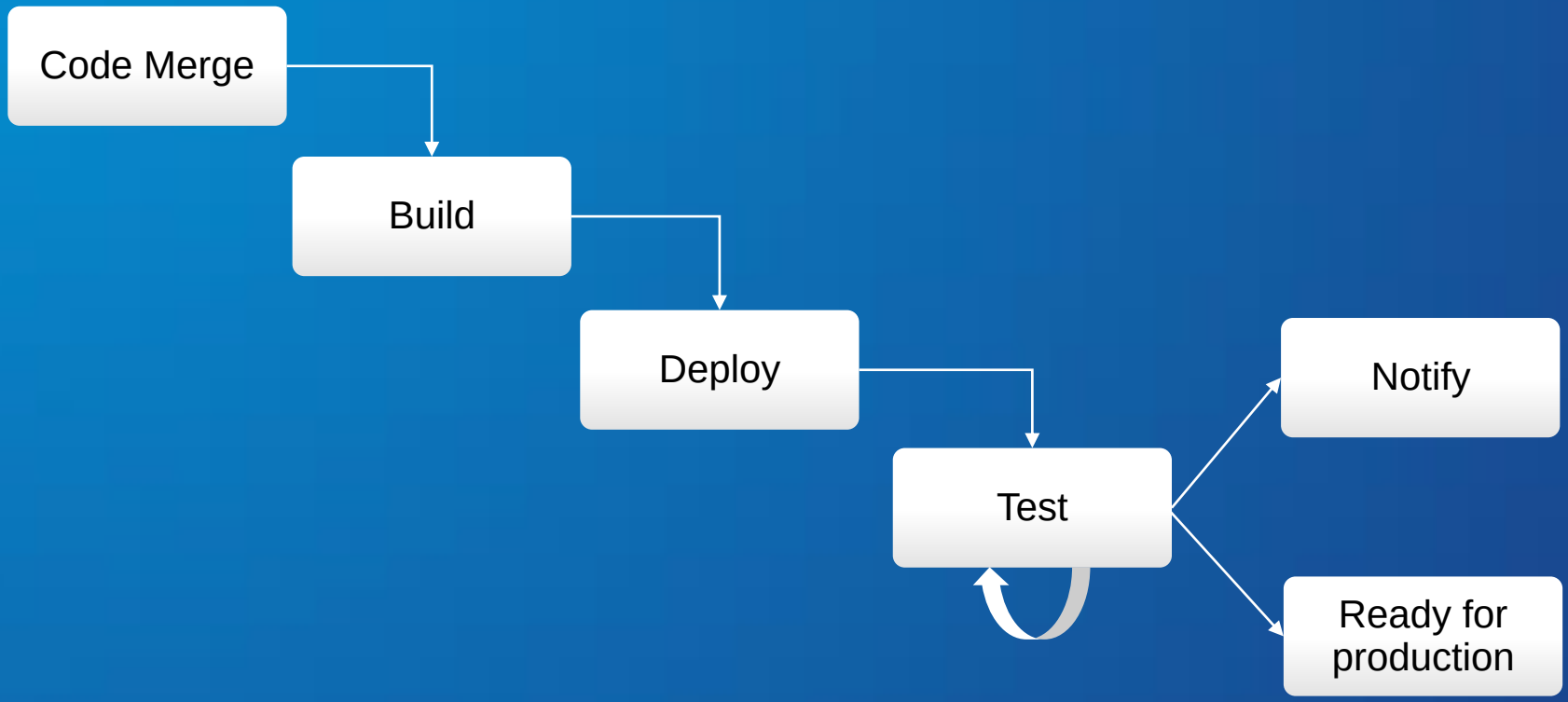
PULL REQUEST PIPELINE



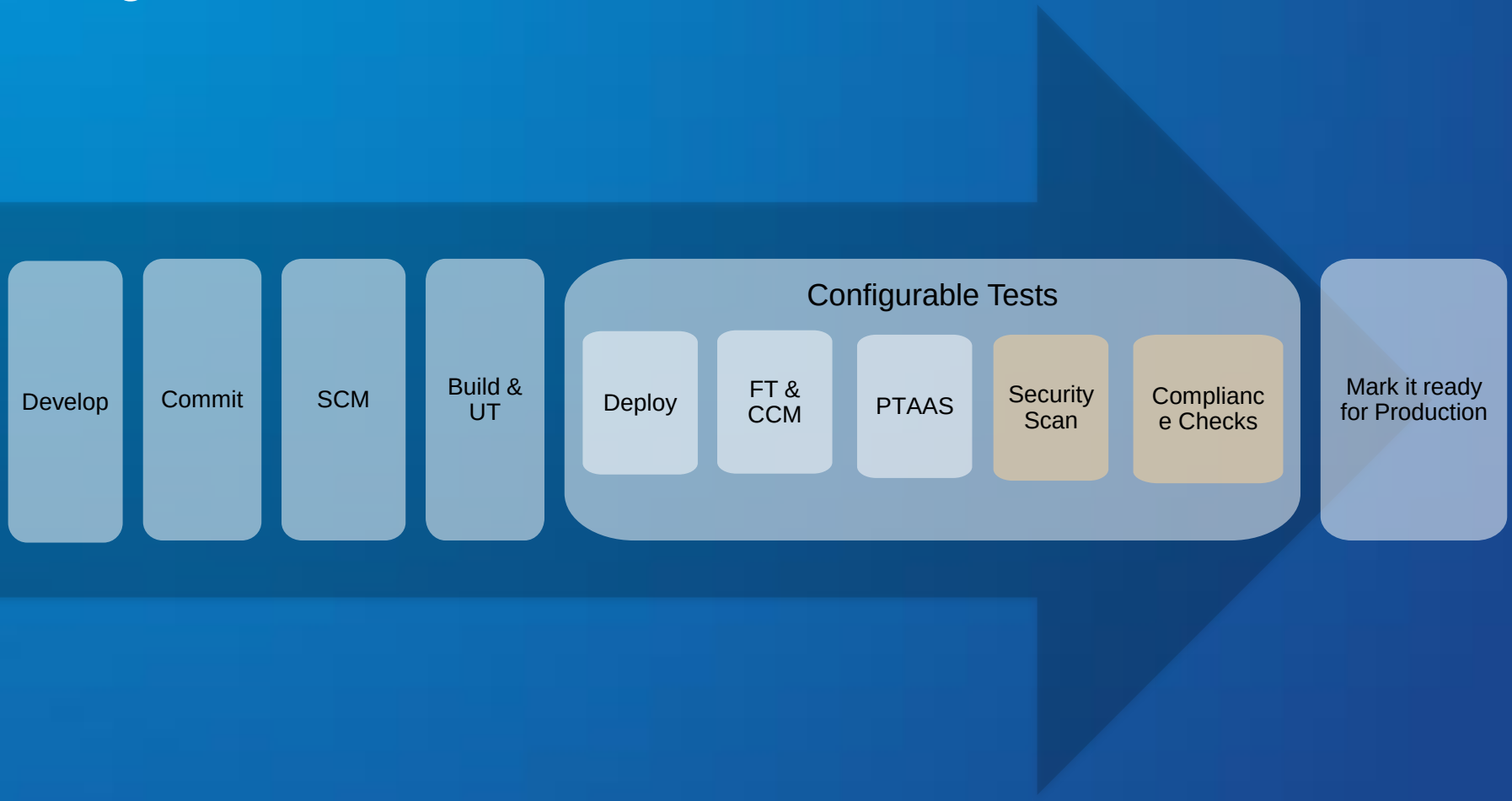
Github PR status!



POST COMMIT PIPELINE



Targeted ECD flow



Thank you!

October 5-6, 2017
Indian Institute of Science, Bangalore
www.modsummit.com

Register early and get the best discounts

April 23-28, 2018
Indian Institute of Science, Bangalore
www.developersummit.com