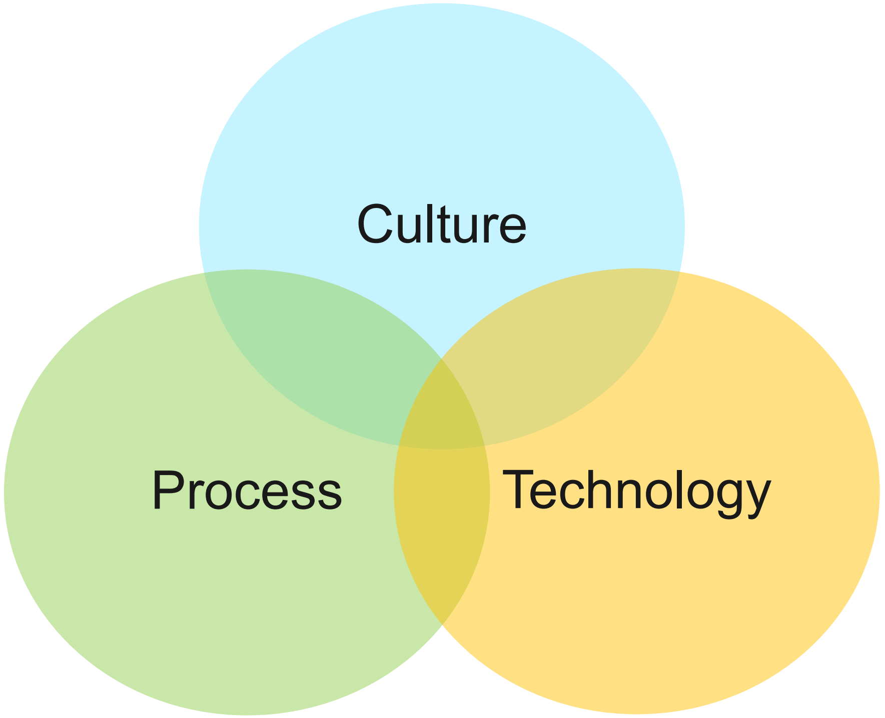
Database as a Service

Db2 for z/OS in a DevOps World

Paul Bartak
Distinguished Engineer
pbartak@rocketsoftware.com

IBM CHAMPION

Before you BUY DevOps, you have to DO DevOps



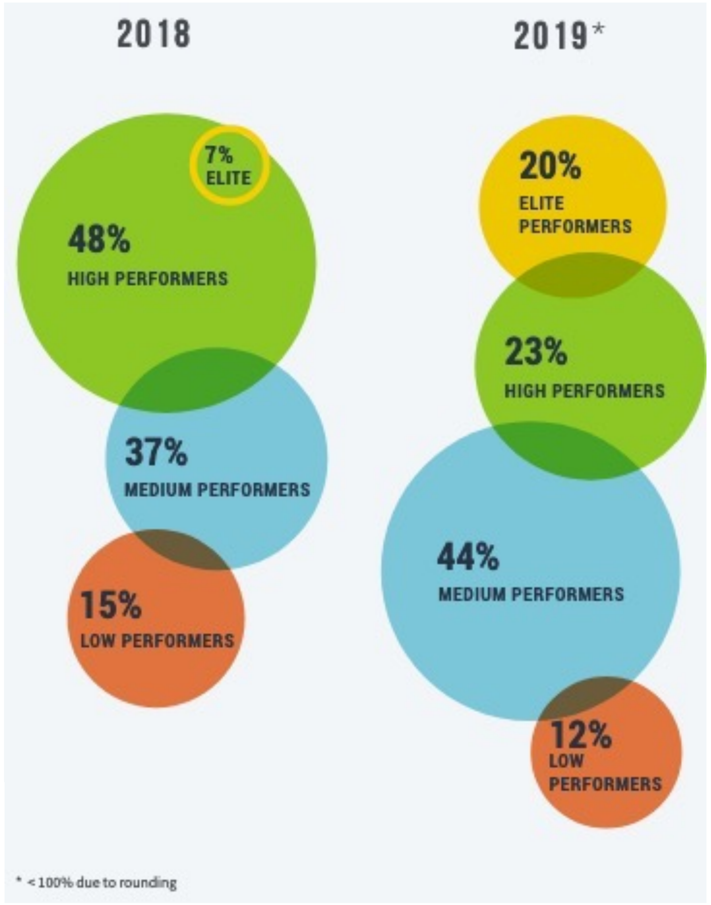
- Efficiency put into practice
- Deal with the change volume, variety, and velocity of digital transformation
- More frequent releases
- Retain / Enhance quality
- Enabling Agile / Lean development
- Fueling continuous integration & delivery

- Requires collaboration and cooperation
- Must be part of the enterprise mission
- Having a seat at the ”Innovation table”

Business Challenges

- The competitive landscape is more challenging than ever
  - Disrupt or be disrupted
  - The Uber Effect – the sharing economy
- DevOps maturity varies but is improving
  - Data Friction is a more recent focus
  - Databases / data sources as Code
- Driving innovation
  - Widespread technology
  - Cloud lowers barriers to entry
- Stopping / reversing downward spirals
  - Core Chronic Conflict
- Innovation delivery
  - Eliminating wait time

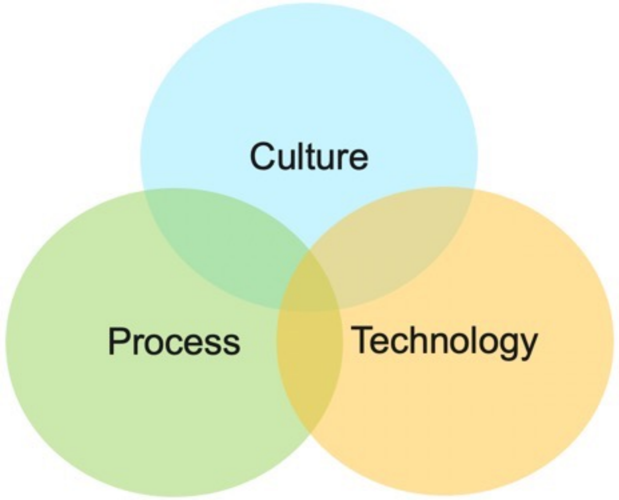
DORA
DEVOPS RESEARCH & ASSESSMENT



What's this all about?

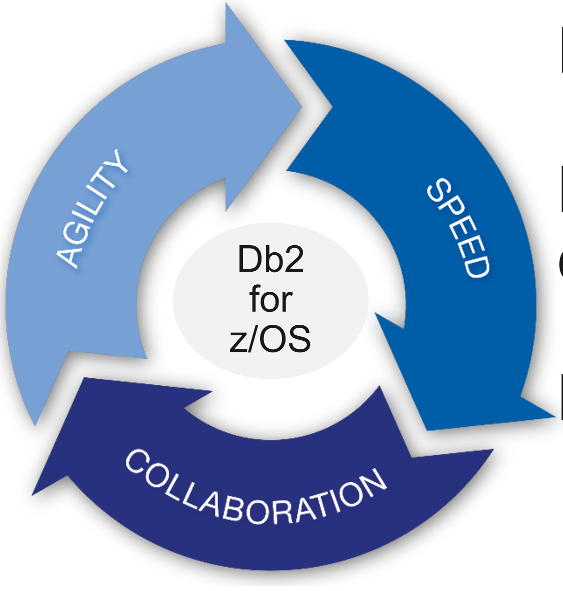
Application **Infrastructure** **Testing** **Build-CI/CD** **Where's Z?** **Where's Db2?**

- **Db2-for-z/OS-Ops**
  - Db2 for z/OS operations at the speed of Developer (remain competitive)
- **DBaaS**
  - The services to enable Db2-for-z/OS-Ops
  - REST services to compose needed automation (or prepackaged for you)
- **Data-sources as code**
  - Extending Infrastructure-as-code to databases
- **Codified rules, thresholds, limits**
  - Guiderails, monitoring, reporting
- **The result** is efficiency (and platform relevance)
  - Elevate the Developer
  - Liberate the Administrator



Db2 for z/OS in DevOps

Brings Db2 applications to market faster with lower costs and less risk



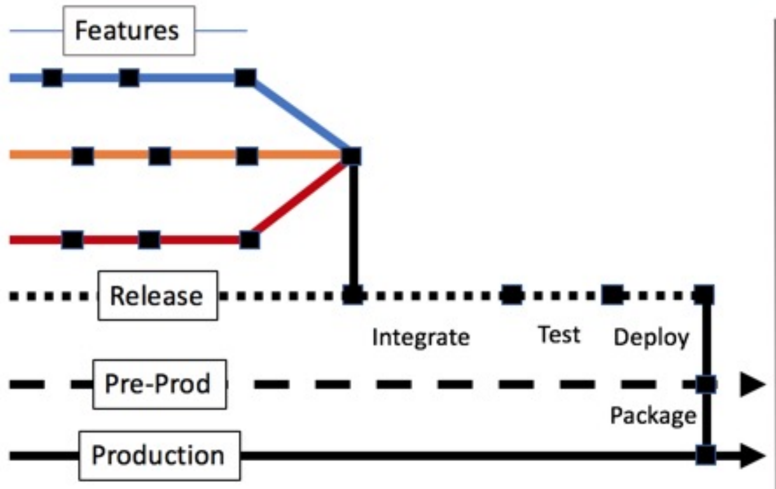
Faster response for the Lines of Business

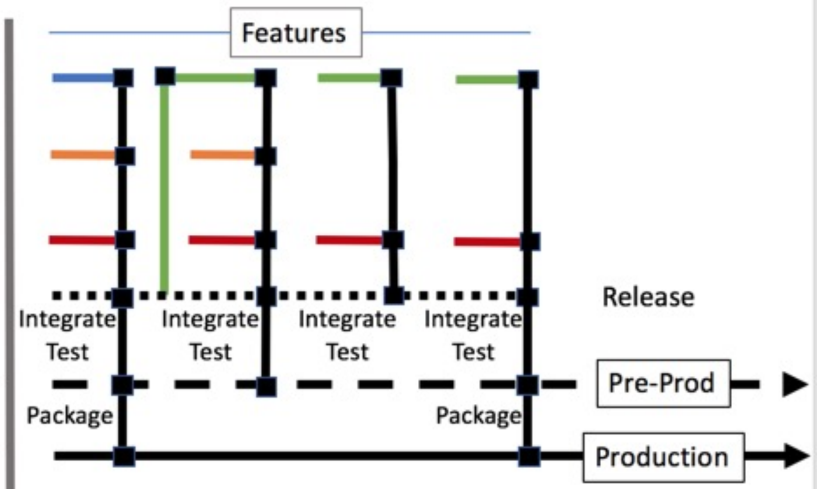
Directives to honor IT / Admin standards & controls

Minimize wait time for Developers

(Wait time is where innovation comes to die)

Move Db2 towards Continuous Integration / Delivery



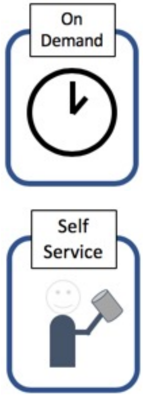


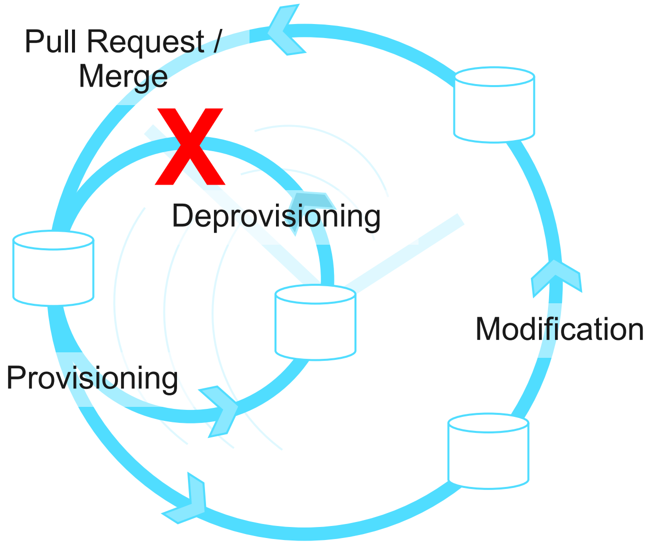
- Large release cycles (months, quarters)
- Slow delivery to customers/marketplace
- Integration is expensive & disruptive
- Problems can have a huge blast radius

- Shorter cycles baked into Dev process
- Faster delivery to customers/marketplace
- Measured / manageable integration
- Contain problems to smaller scopes
- Lower stress associated with release delivery

What about the Db2 assets? Db2 **DBaaS**

On-demand, Self-service, Developer Driven

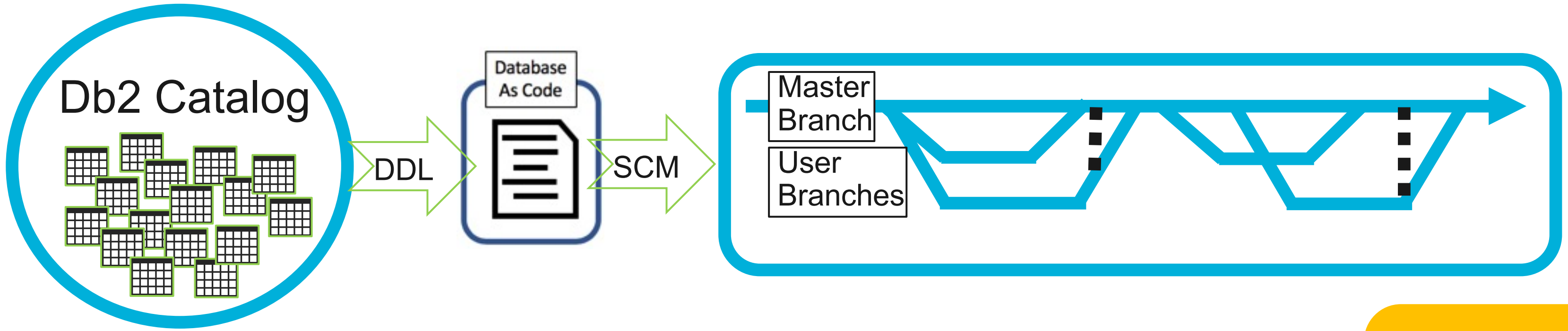




Elevate the Developer

- Drive Database needs in the Developers cadence by the Developer
- Provision an Instance as needed within the Sprint
- Fail fast, Deprovision the Instance and (perhaps) try again
- Deploy changes to the Instance as needed
- Can submit changes for consideration to include in the master branch
  - Pull Request

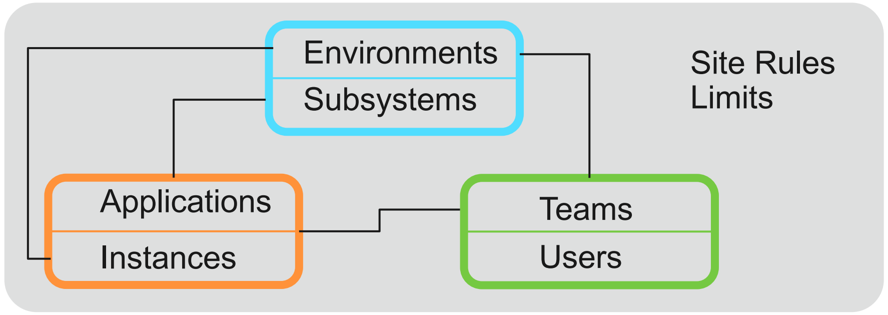
Database (DDL) as Code



Liberate the Administrator

- Database as code (versioned DDL):
  - Logical groupings of Db2 objects (in support of Applications)
- Unites with:
  - Application version control
  - Infrastructure as code
- Fuels provisioning request & change deployments

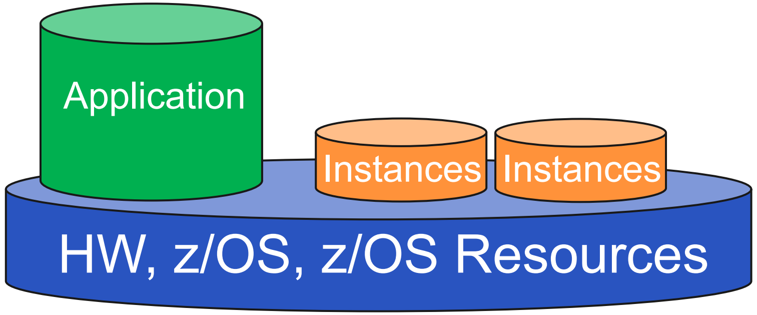
Management/Administrative Directives

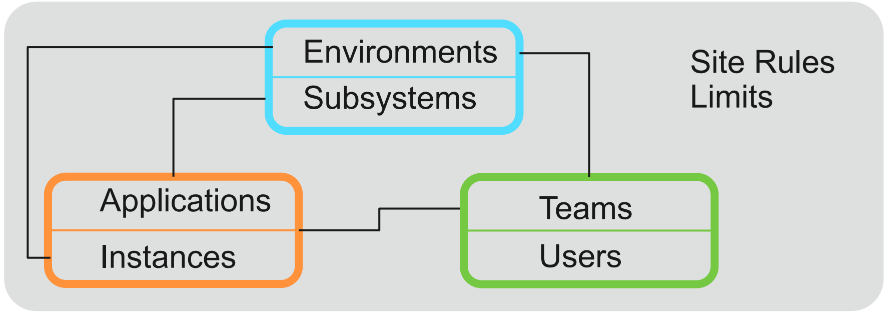


- Environment definitions to control where Provisioning takes place
- Provisioning Instance Limits
- Administration of Application via Teams
- Storage Limits monitoring Teams, Applications, Users, and Environments
- Site Rules for naming, definitions, placement
- Data Steward roles for approving database changes

Liberate the Administrator

DevOps In a Shared Environment



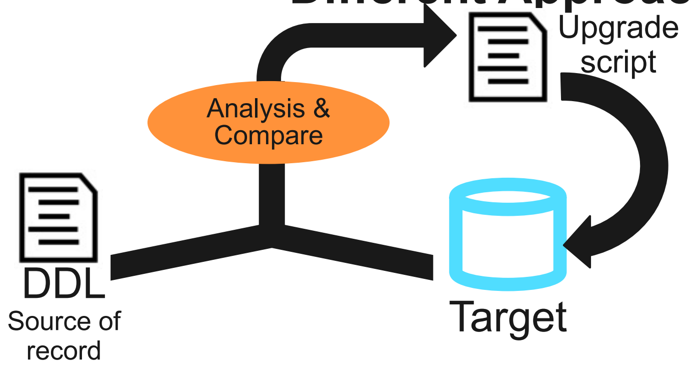


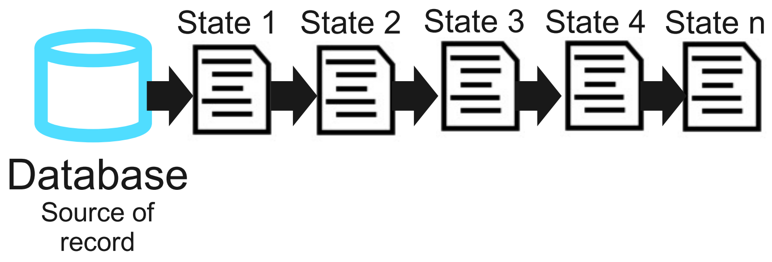
- Distributed environments provisioning can be distinct from infrastructure, up
- This could also be the case with z/OS with a virtual environment (zD&T)
- But it's more likely that the HW, z/OS, and z/OS resources (Db2, storage, etc.) will be shared
- Important elements of DevOps in a shared environment:
  - Registration of participating Db2s & Db2 objects
  - Control where provisioning activities will take place with limits:
    - Expanded, fenced authorities for Developers
  - Namespace management for Instance separation
  - Rules for naming, placement, definitions
  - Storage monitoring
  - Easy visibility to rules, metrics, etc.

Modernize the Platform

State-based vs. Migration-based Approaches

Different Approaches to Database (DDL) as Code





State Based

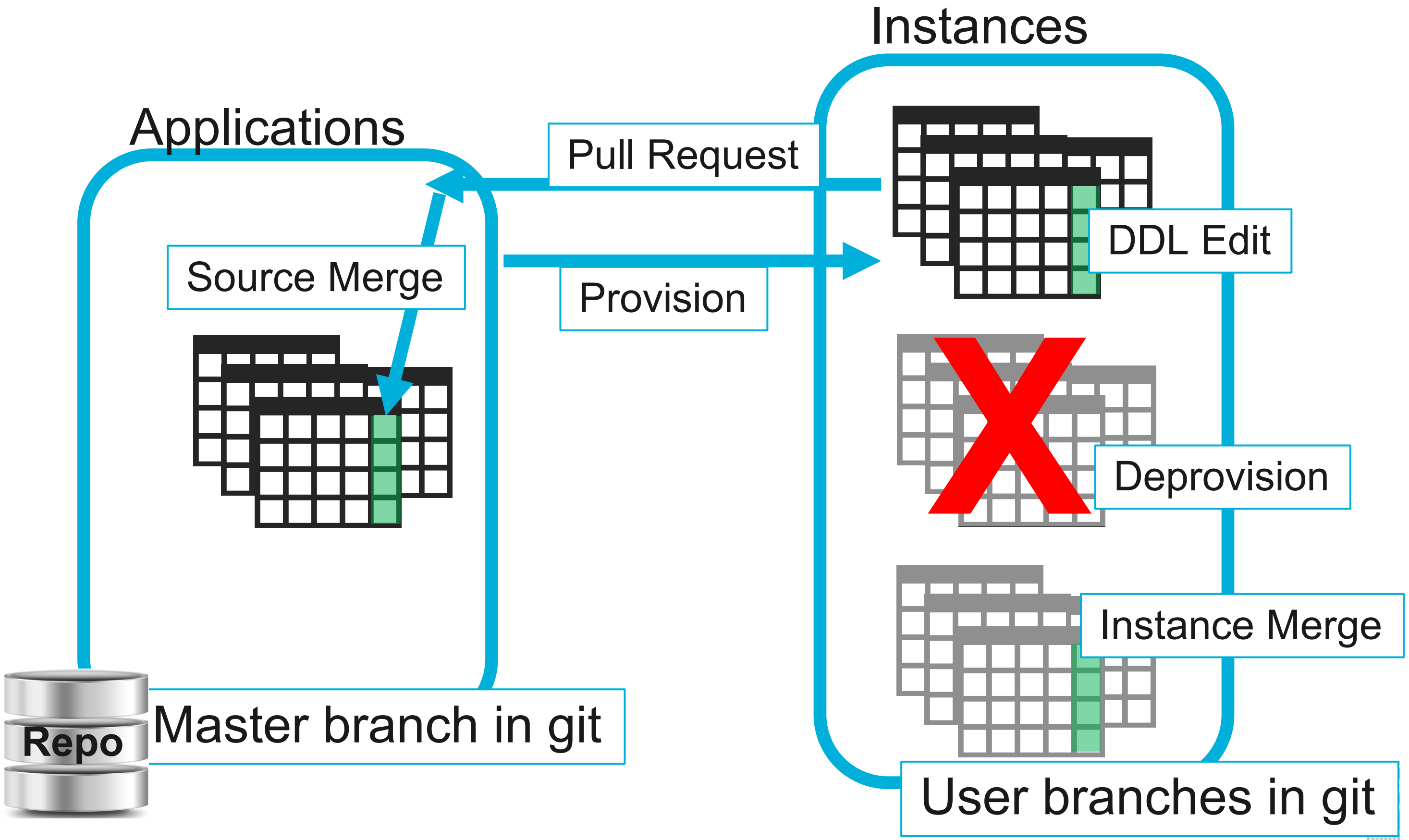
- Source control system of record
  - Established from snapshot of DB
- DDL stored as version control text files
- Has a Compare engine
  - Indicate desired state
  - Engine optimizes change for target

Migration Based

- Database system of record
- Capture state at beginning of project
- Maintain series of sequenced migration scripts
- Use culmination of scripts to achieve desired state

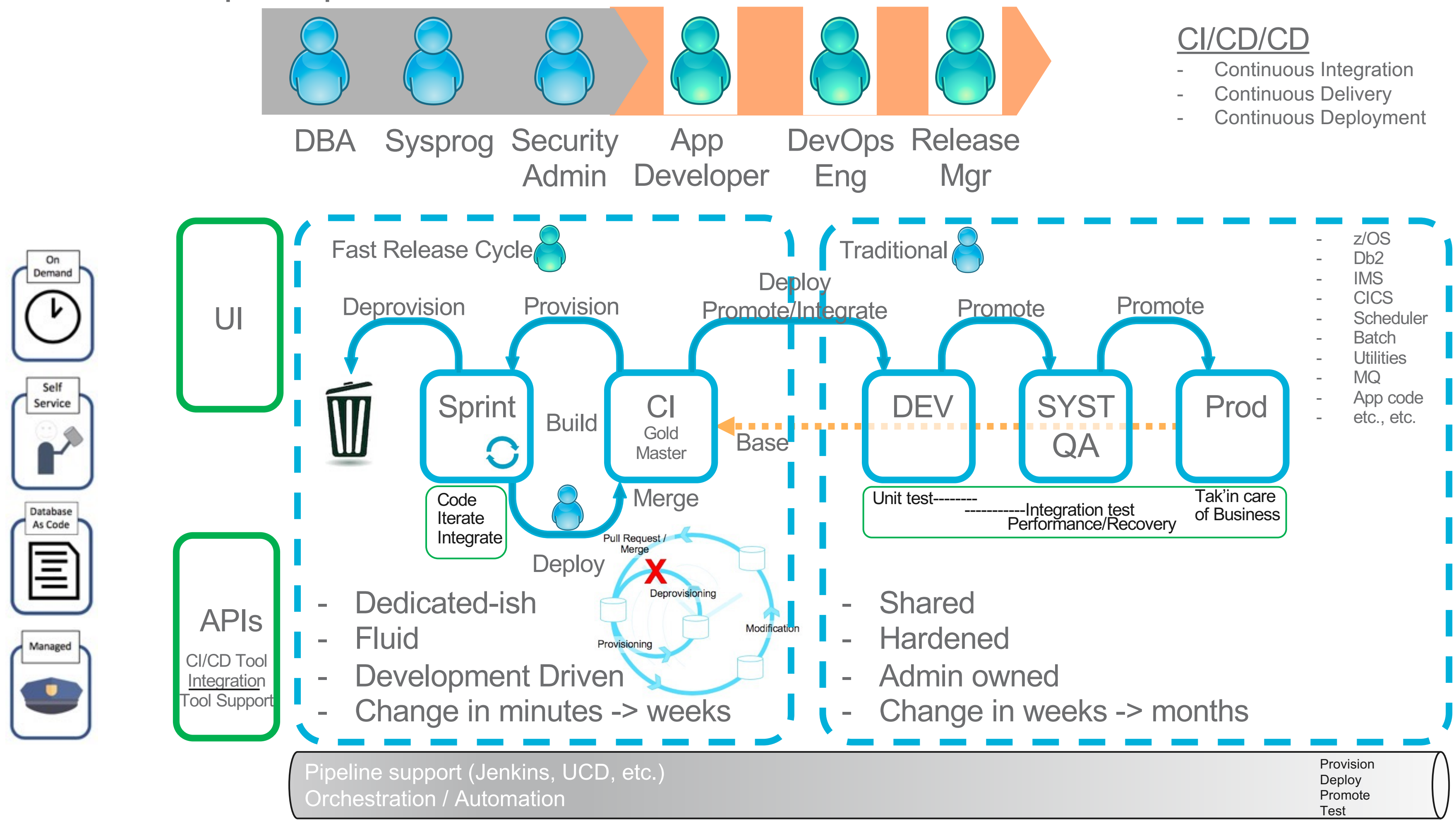
Db2 DevOps Example Flows

- Subsystems are registered
- Users, Teams set up and assigned Environments
- Site Rules defined



IBM Db2 DevOps Experience for z/OS Whiteboard

<u>CI/CD/CD</u>
- Continuous Integration
- Continuous Delivery
- Continuous Deployment

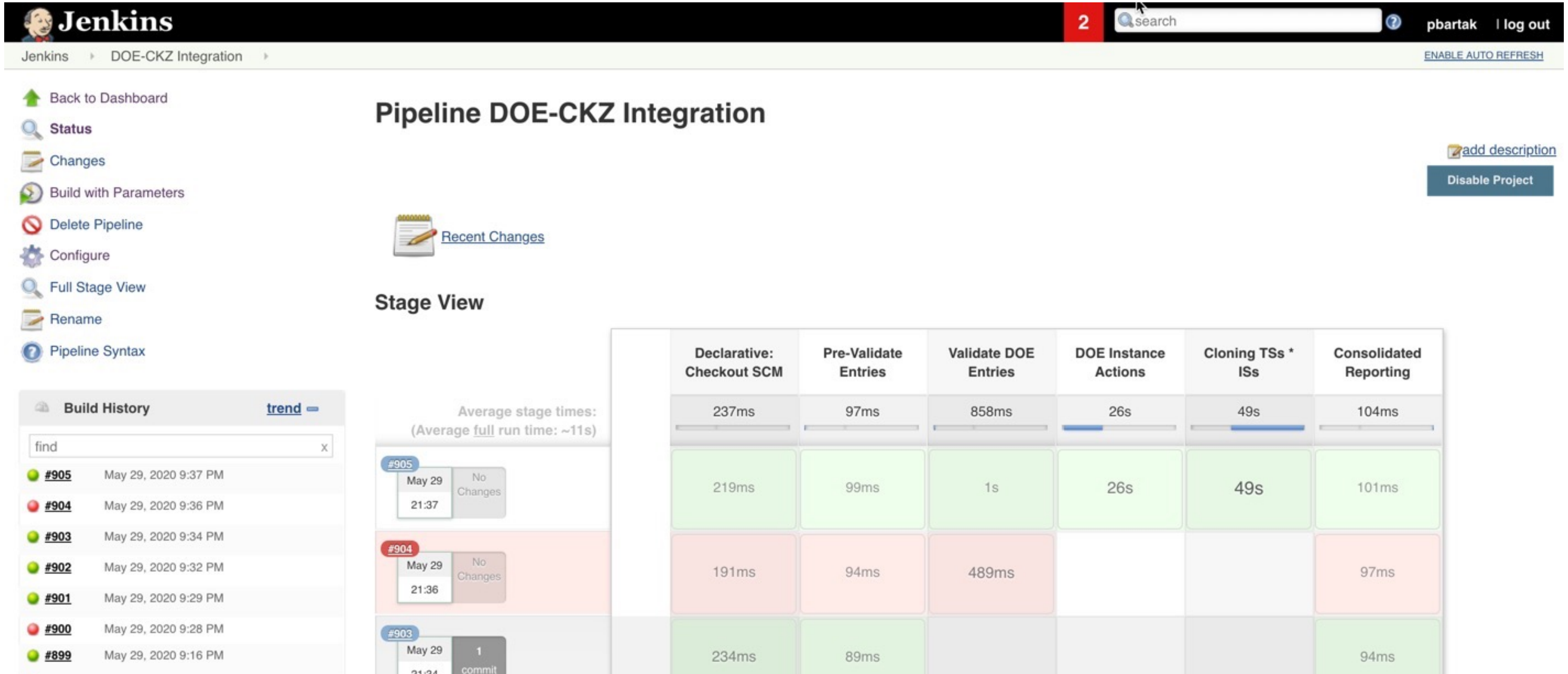


- Dedicated-ish
- Fluid
- Development Driven
- Change in minutes -> weeks

- Shared
- Hardened
- Admin owned
- Change in weeks -> months

Db2 for z/OS Ops – CI/CD Integration

- Sample Jenkins pipelines have been developed for:
  - Provisioning
  - Deprovisioning
  - Reprovisioning
  - Instance update
- This model can be extended to other use cases or CI/CD tools



Pipeline DOE-CKZ Integration

Build History — trend

Build	Date
#905	May 29, 2020 9:37 PM
#904	May 29, 2020 9:36 PM
#903	May 29, 2020 9:34 PM
#902	May 29, 2020 9:32 PM
#901	May 29, 2020 9:29 PM
#900	May 29, 2020 9:28 PM
#899	May 29, 2020 9:16 PM

Stage View

	Declarative: Checkout SCM	Pre-Validate Entries	Validate DOE Entries	DOE Instance Actions	Cloning TSs * ISs	Consolidated Reporting
Average stage times: (Average full run time: ~11s)	237ms	97ms	858ms	26s	49s	104ms
#905 May 29 21:37 No Changes	219ms	99ms	1s	26s	49s	101ms
#904 May 29 21:36 No Changes	191ms	94ms	489ms			97ms
#903 May 29 21:34 1 commit	234ms	89ms				94ms

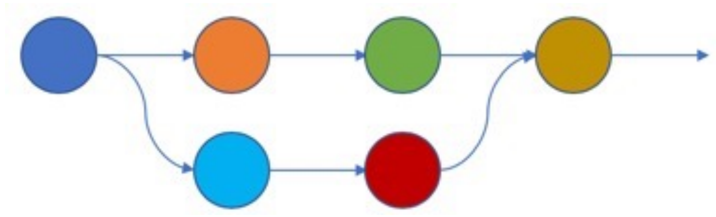
Pipelines

- There are many open source & commercial pipelines available
- Workflows orchestrate services much like a scheduler orchestrates job streams
  - REST calls
  - Shell scripts
  - Templates
- Scripting languages customize the experience
- Declared pipelines create Pipelines-as-Code
  - The pipeline code managed under version control
  - The pipeline tool checks out the pipeline code and runs it.

Jenkins

- There are many open-source options, but Jenkins tops most CI/CD lists
- There are 1000s of plug-in options to customize the experience
- Pipeline syntax
- Groovy is the scripting language
- Dashboard for managed workflows
- Simple UI to accept variables into workflows
- Can use a webhook for “headless” workflow initiation
  - Use JSON payload to pass the variables
- Keeps a history of pipeline execution & performance

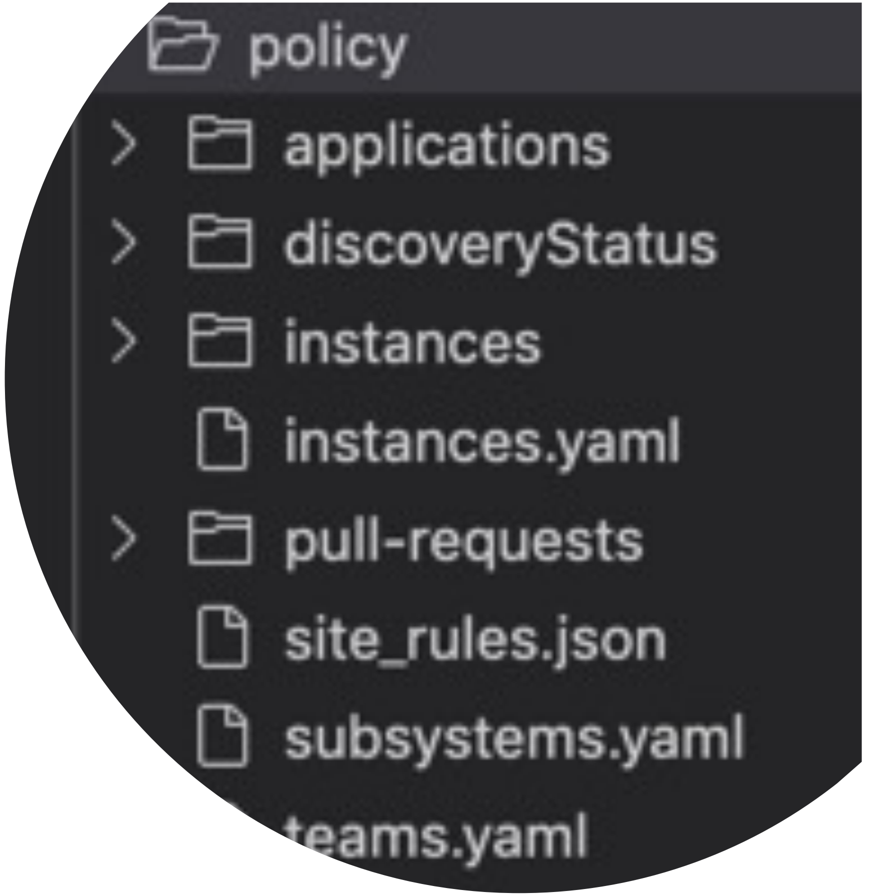
Things to consider when composing APIs



- **Metadata management**
  - What is the source for the input to services?
  - Will you require inputs from the invoker?
- **Synchronous vs. Asynchronous**
  - Consideration for modification APIs (POST, PUT, PATCH, DELETE)
  - Many modification APIs are ”fire and forget” (asynchronous), returning a result before the work is done
  - Will your pipeline tolerate this behavior?
- **Polling (for asynchronous APIs)**
  - To validate that the API work has been completed
  - Looping on an associated GET call to check completion status
- **Getting started**
  - Check for documentation (Swagger)
  - Use a curl-based tool (like Postman) to “unit test” the service and validate expected results

Metadata & DBaaS

- **DevOps adoption**
- **DBaaS object definitions**
  - What is available & How is it being used
  - Are the constructs easy for a Developer to consume
- **Ownership & editability**
- **Approval cycle**
  - Review and integration
  - Drive toward schema synchronization
- **Monitoring & usage**
  - **Control for Administrators**



Thank You