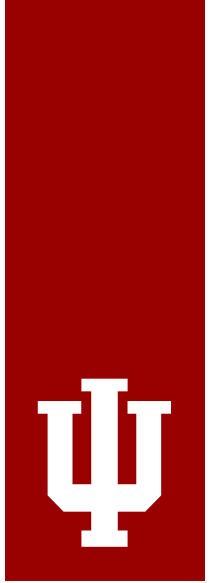


INDIANA UNIVERSITY
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

Science Gateway Architectures

August 21st 2018

Suresh Marru, Marlon Pierce

Todays Outline

- Class Logistics
- Motivation & Goals
- Course Overview
- Open Discussion

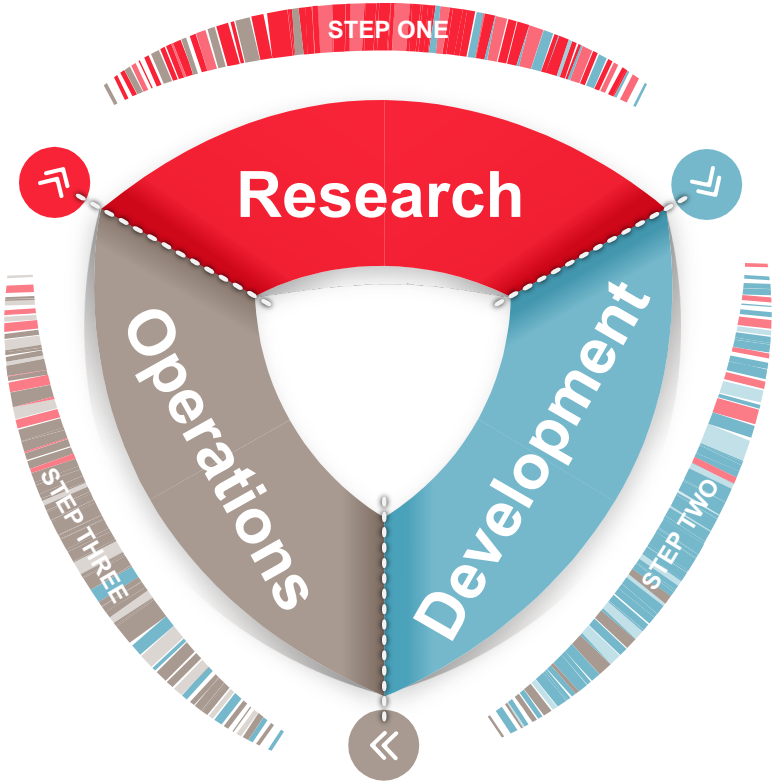
Class Introductions

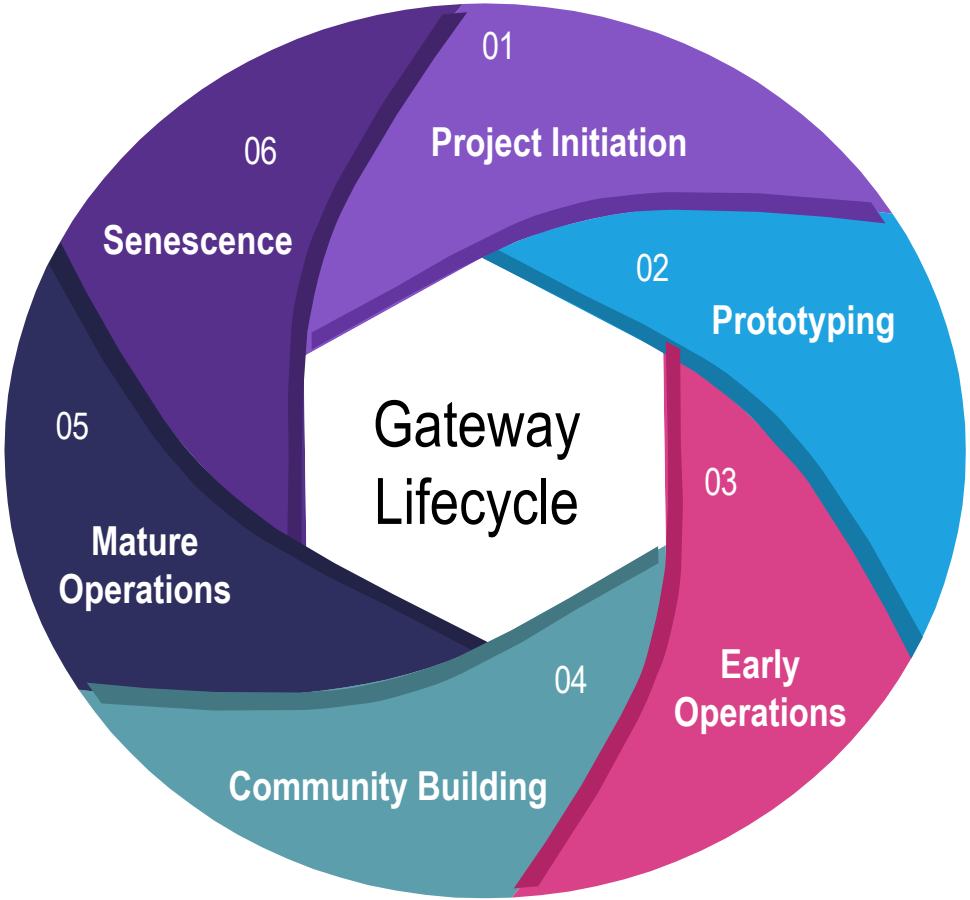
Share your goals, expectations, concerns and a brief background

Science Gateways Research Center

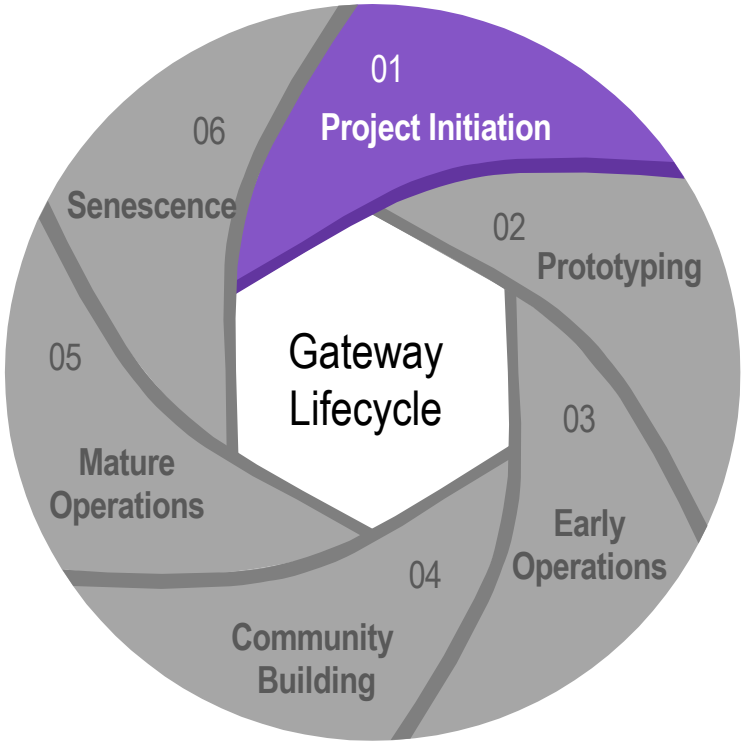
Accelerate research, discovery and scientific collaboration through the creation, integration, and operation of systems that combine distributed software, data, and infrastructure resources into user environments that serve and enable scientific communities. Advance the broader cyberinfrastructure community through education, training and sharing knowledge.

Lifecycle of typical Science Gateway




Gateway Lifecycle
01
Project Initiation
02
Prototyping
03
Early Operations
04
Community Building
05
Mature Operations
06
Senescence

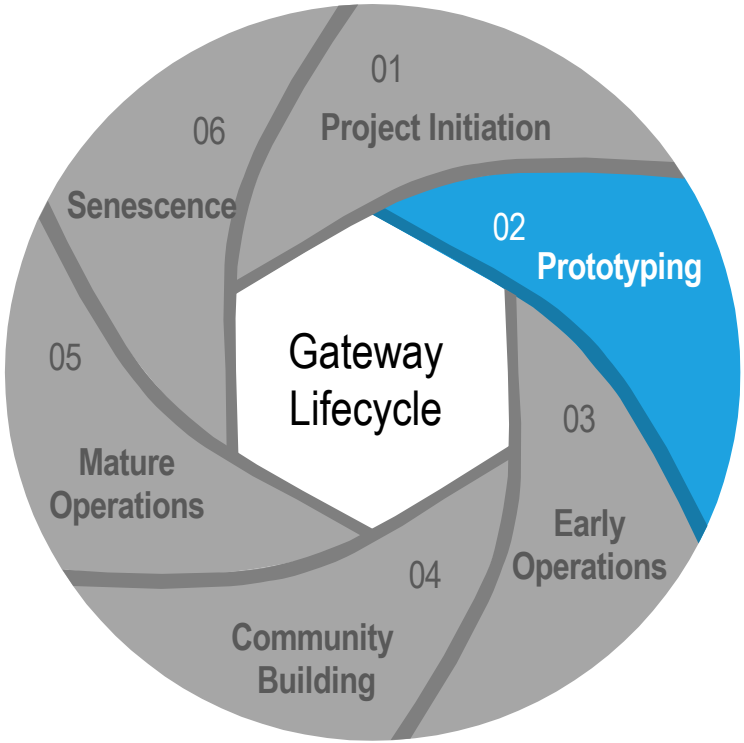
Project Initiation: you want to build a gateway



Technical Challenges

- Establishing your technical base
- Choosing technologies that maps your value proposition.
- Assemble team: Who do you need

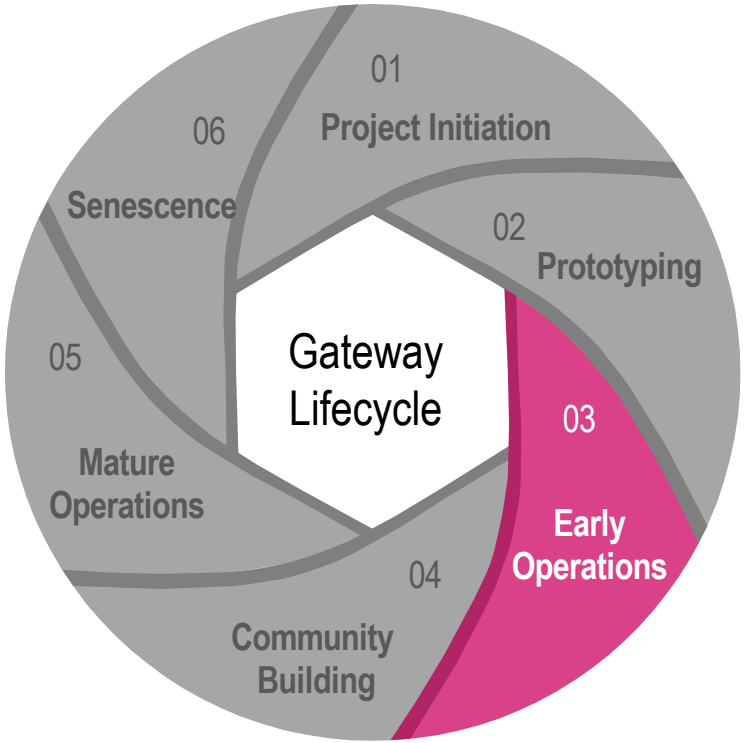
Prototyping:
Choosing a framework vs. building things yourself



Technical Challenges

- Evaluating Frameworks: How well can they be adapted to what you want?
- DIY: assembling the right team, establishing the right engineering and operations practices
- DIY: choosing third party systems

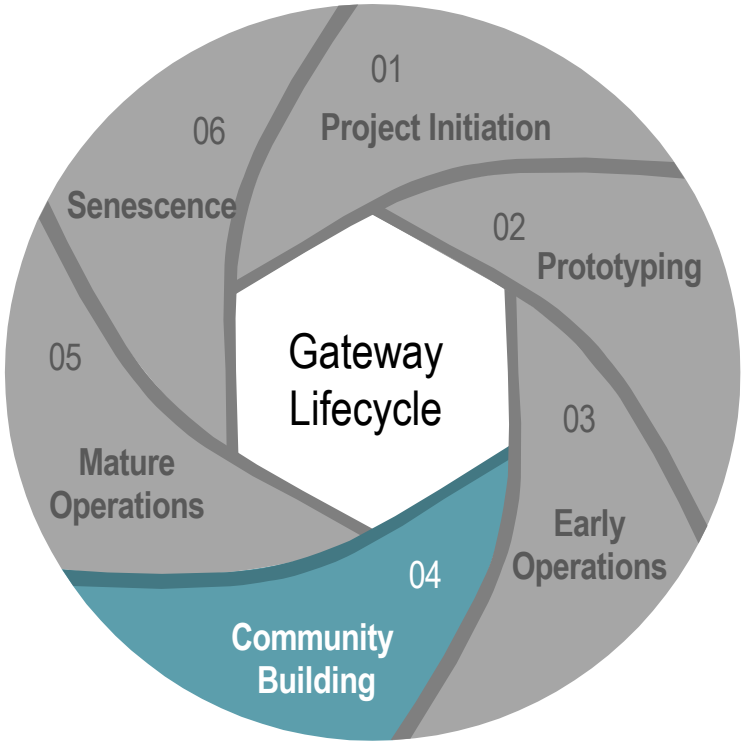
Early Operations:
Transitioning porotype into “production”



Technical Challenges

- High Availability
- Scaling up your technology for more users
- Monitoring
- Loss of key tech people; Onboarding new developers

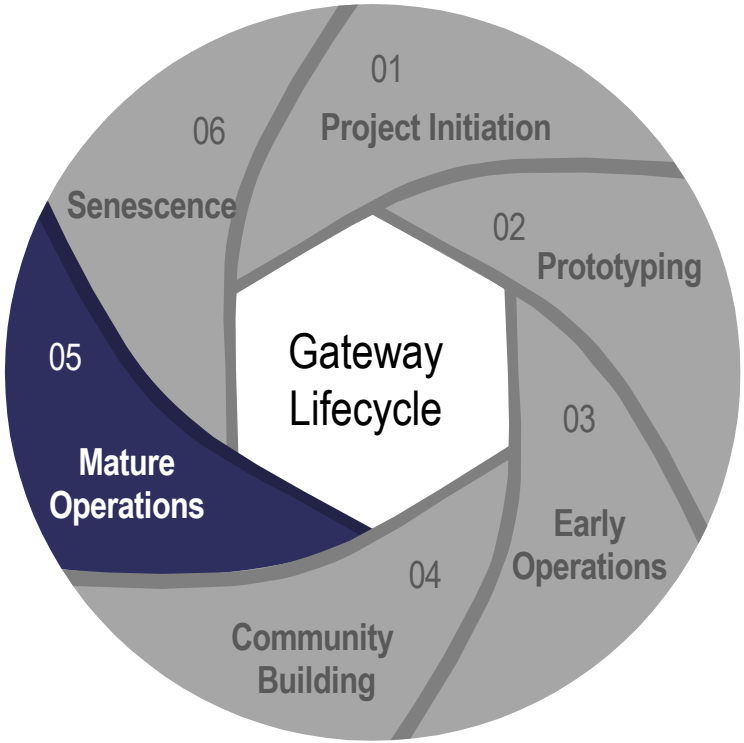
Community Building:
Have a technology but need to grow your user base



Technical Challenges

- Adding features expeditiously
- Managing feature creep

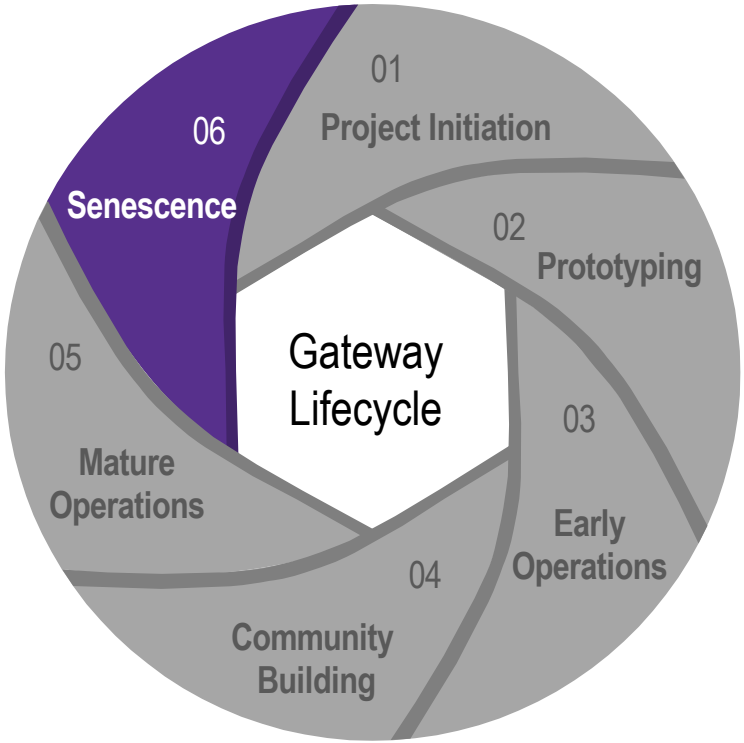
Mature Operations:
Mature in-house systems + Community



Technical Challenges

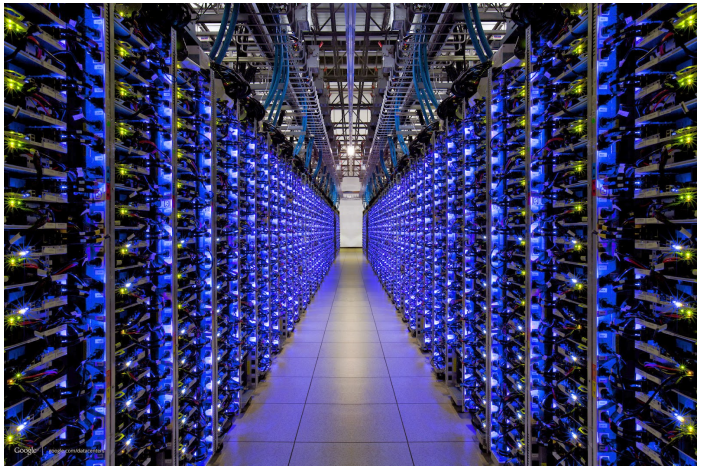
- Technical Debt: dealing with earlier short cuts and shortcomings
- Expanding your team's roles
- Difficulties changing system to take advantage of new technology
- Loss of key people, onboarding
- Plateaued usage

Senescence:
Aging technology, value proposition may need revision



Technical Challenges

- Wholesale tech stack change
- Back to square one



Data Centers

Brainstorm a case study

Pick a “Software as a Service” example which you cannot live with being offline

Source: https://www.facebook.com/note.php?note_id=469716398919

Class Objectives

- Provide a high level, broad understanding of the application of core distributed computing systems concepts.
- Study both abstract concepts and practical techniques.
- Understanding state of the art and apply the general concepts of Distributed Systems in developing a science gateway.
- Open source philosophies modelled after Apache Software Foundation.

Distributed Systems

"A distributed system is a collection of entities, each of which is autonomous, programmable, asynchronous and failure-prone, and which communicate through an unreliable communication medium."

- Prof. Indranil Gupta, UIUC

Common Distributed Characteristics

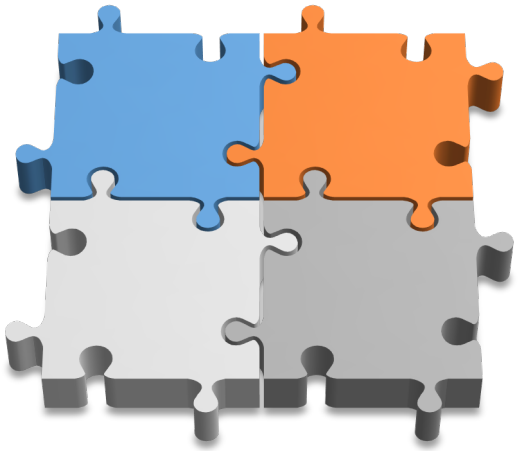
- Heterogeneity
- Robustness
- Availability
- Transparency
- Concurrency
- Efficiency
- Scalability
- Security
- Openness

Course Outcomes

- Understanding of microservice architectures and their underlying distributed systems foundations.
  - Componentization & Containerization.
  - Scalability & Resilience
  - Monitoring & Analytics
- Applied understanding of the DevOps principles of continuous integration and delivery to the development and operations.
- Understanding of open source practices, particularly those of the Apache Software Foundation.
- Understanding of Science Gateways.

Project Expectations: Continuous Integration

Continuous integration is a development practice that requires developers to integrate code into a shared repository on a daily basis.



Each check-in is validated by

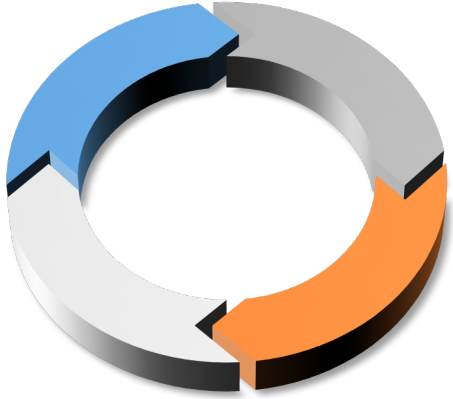
- An automated build
- Automated unit, integration and acceptance tests

Integrating regularly in production-like environments makes it easier to quickly detect and locate conflicts and errors.

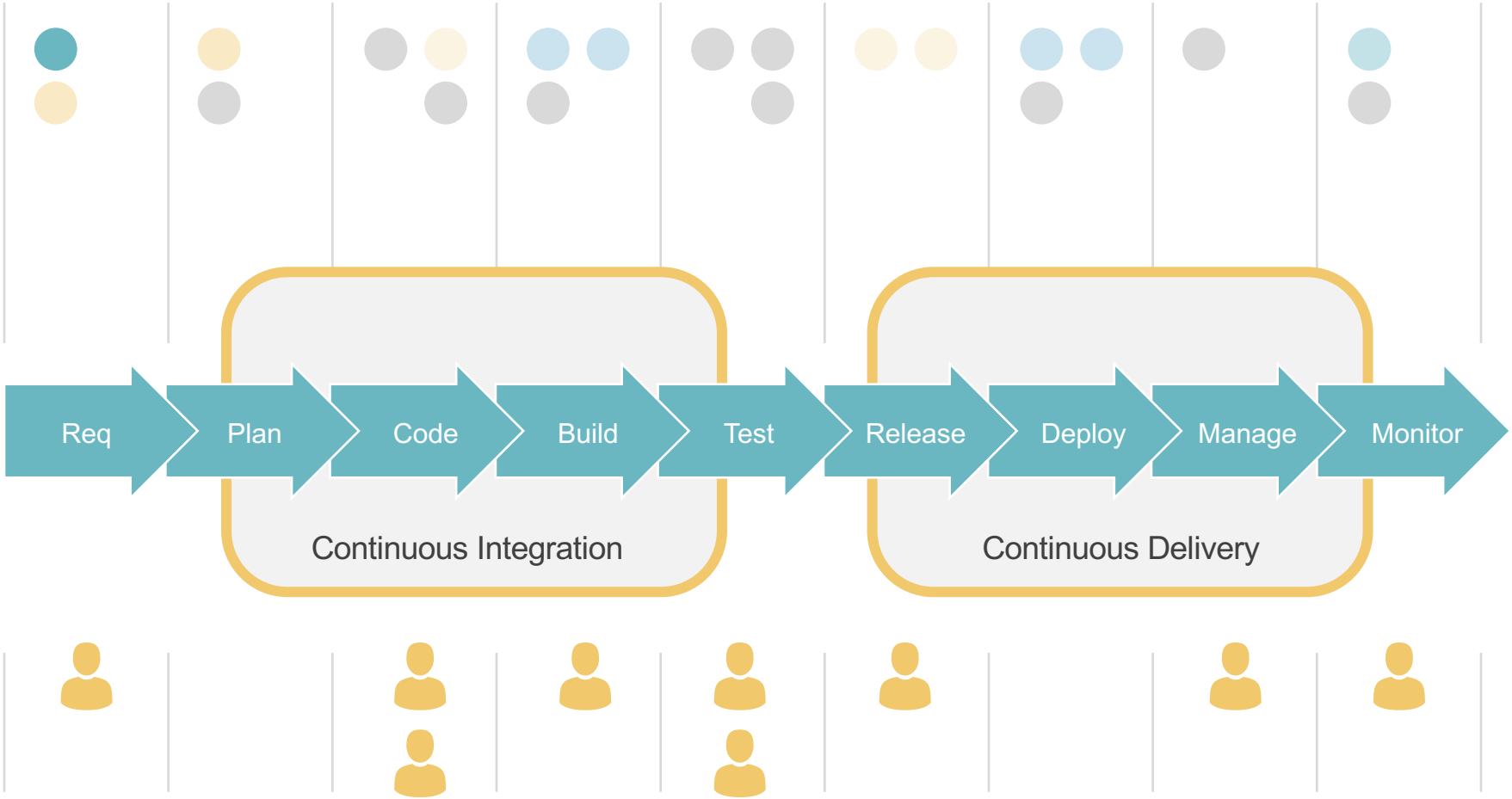
Project Expectations: Continuous Delivery

Continuous delivery is a methodology that focuses on making sure software is always in a releasable state throughout its lifecycle.

- Extends continuous integration
- Provides fact, automated feedback on the production-readiness of systems
- Prioritizes keeping software deployable over working on new features
- Enables push-button deployments on demand
- Reduces deployment risks and enables quicker user feedback



Integrating regularly in production-like environments makes it easier to quickly detect and locate conflicts and errors.


Req
Plan
Code
Build
Test
Release
Deploy
Manage
Monitor
Continuous Integration
Continuous Delivery

Course Projects

- There will be 4 project milestones, each one is worth 20 points.
  - Project Milestone 1: Microservices with DevOps
    - Due Late September
  - Project Milestone 2: Security, Auditing, Distributed Coordination
    - Due Late October
  - Project Milestone 3: Reliability & Scaling
    - Due Late November
  - Project Milestone 4: Apache Airavata
    - Due Finals Week
- Mid-term and finals are project demonstrations, each one worth 5 points.

Lecture Topics

- Performance
  - Caching
  - Load imbalance
- Data persistence
  - SQL vs NoSQL
  - Data models
  - CAP Problem
- Communication
  - Push or Pull
  - Queues
  - Proxies
  - Long Polling
  - Web Sockets

Lectures Contd.

- Fault Tolerance
  - Active
  - Passive
- APIs
  - Design
  - Versioning
- Data Partitioning (AKA sharding)
  - Horizontal Partitioning
  - Vertical Partitioning
  - Directory based partitioning

Case Studies

- Apache Airavata

Advance Track

Should be able to all of basic track + more

Volunteers?