A Brief Survey of the Web Deployment Landscape

Caleb Gosnell

Python Web Conference 2020

Expectations

This talk might help if you are:

- new
- overwhelmed or confused
- feeling the fear of missing out

Agenda

➡️ Preliminaries

- A General Model
- Catalog of Practices
- Example Workflows
- Considerations, Caveats, and Advice

Caleb Gosnell

DevOps Engineer at
Six Feet Up

Trying hard to manage computers and networks, and deploy other people's code since 2008.

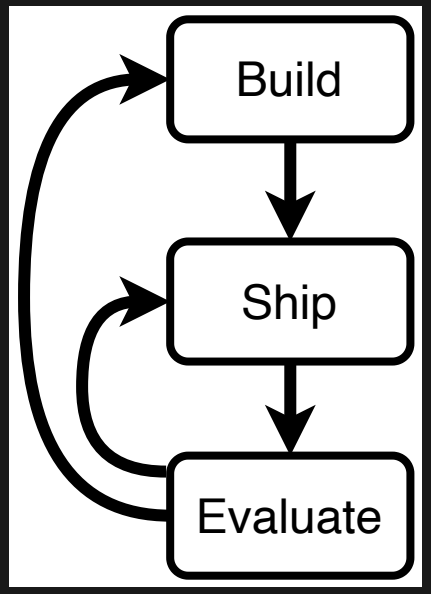
Agenda

✅ Preliminaries

➡️ A General Model

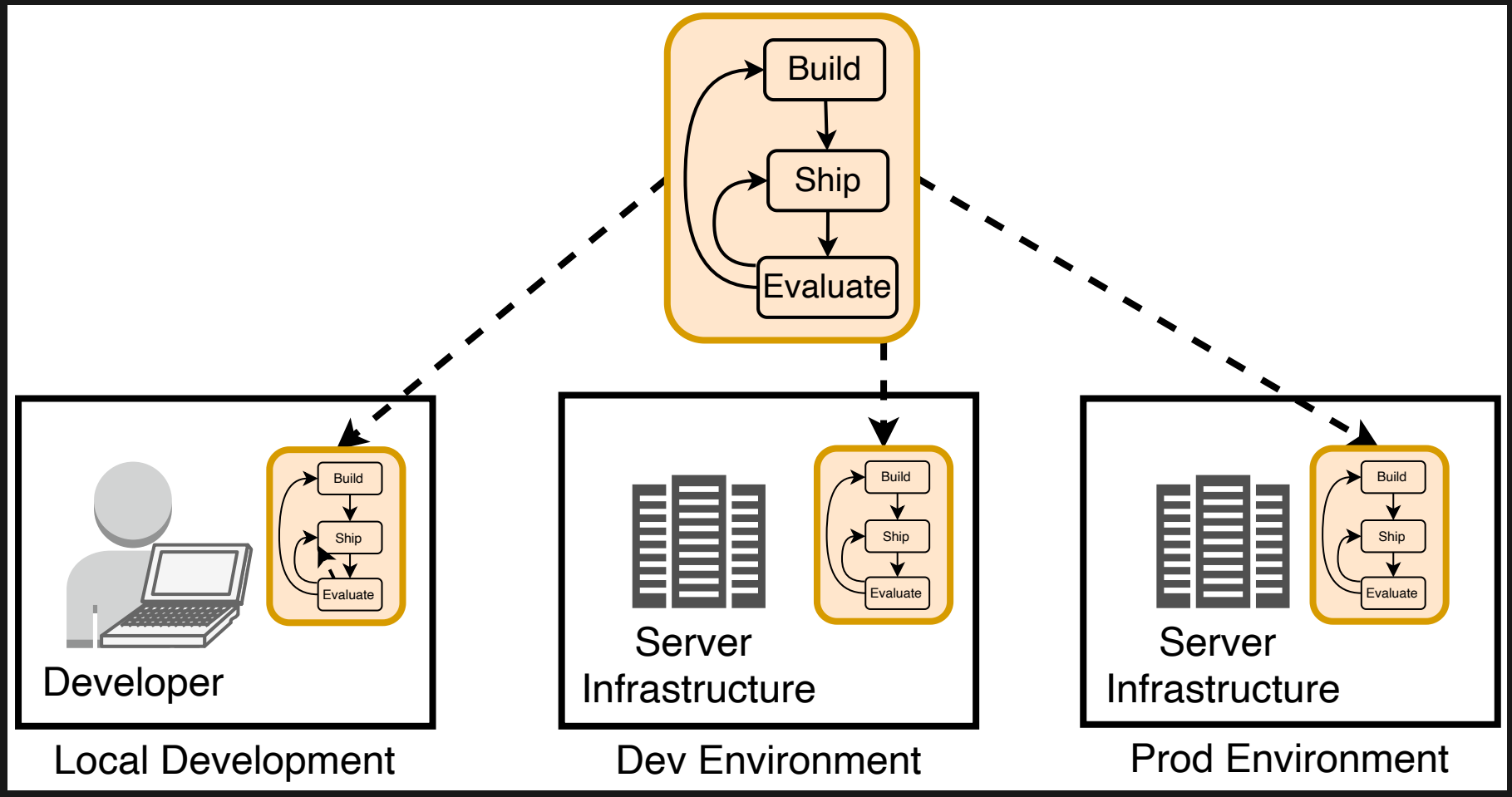
- Catalog of Practices
- Example Workflows
- Considerations, Caveats, and Advice

Model Deployment Steps



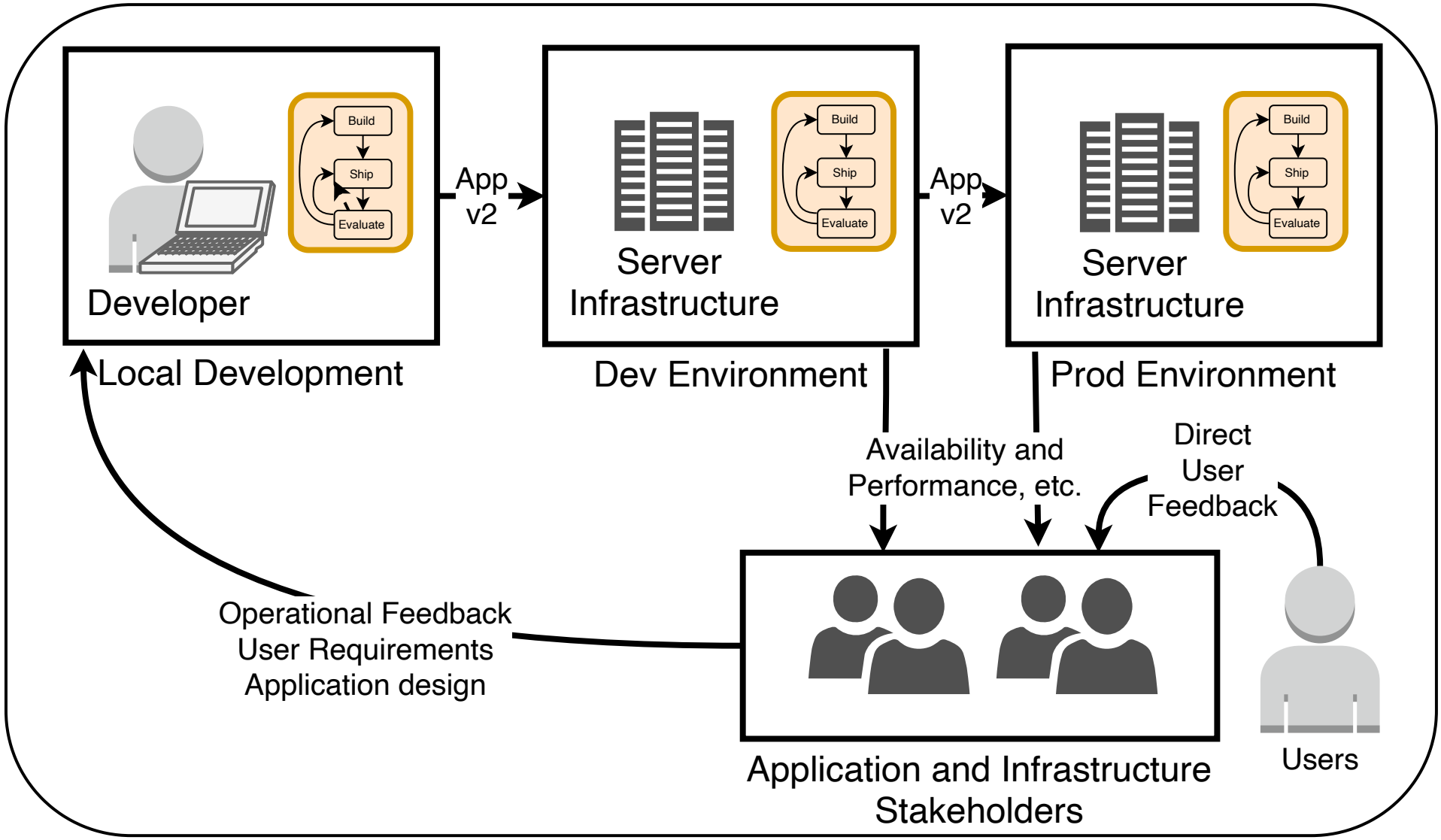
Build: produces a release/artifact
Ship: launches a release
Evaluate: collects information resulting from the release

Everything is a Deployment



Deployment Context - SLDC

Software Development Lifecycle

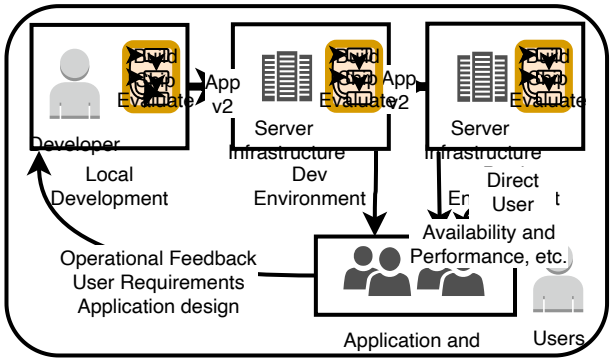


SLDC Context - your Organization/the Universe

Normal Spacetime

Your Organization

Software Development Lifecycle



SLDC of other Apps

Org policy, procedures

Other initiatives, opportunities

Everything Else

Agenda

- ✅ Preliminaries
- ✅ A General Model
- ➡️ Catalog of Practices
  - ➡️ Build
    - Ship
    - Evaluate
- Example Workflows
- Considerations, Caveats, and Advice

The Build Step

- Usually executes on a CI server or Repository service
- With a nice IDE, it can start while the code is still being written
- locally at commit time

Build Practices

- static analysis: style linting, error detection, automated refactoring
- dependency resolution
- vulnerabily checking
- execute unit tests
- manual code review
- semantic versioning
- packaging: eggs, wheels, docker containers, golden VM images
- distribution: upload the release/artifact to a package service or artifact store

Agenda

- ✅ Preliminaries
- ✅ A General Model
- ➡️ Catalog of Practices
  - ✅ Build
  - ➡️ Ship
    - Evaluate
- Example Workflows
- Considerations, Caveats, and Advice

The Ship Step

Can execute from:

- a CI server
- repository service
- configuration management service
- an infrastrcture provider
- or locally

Ship Step Practices

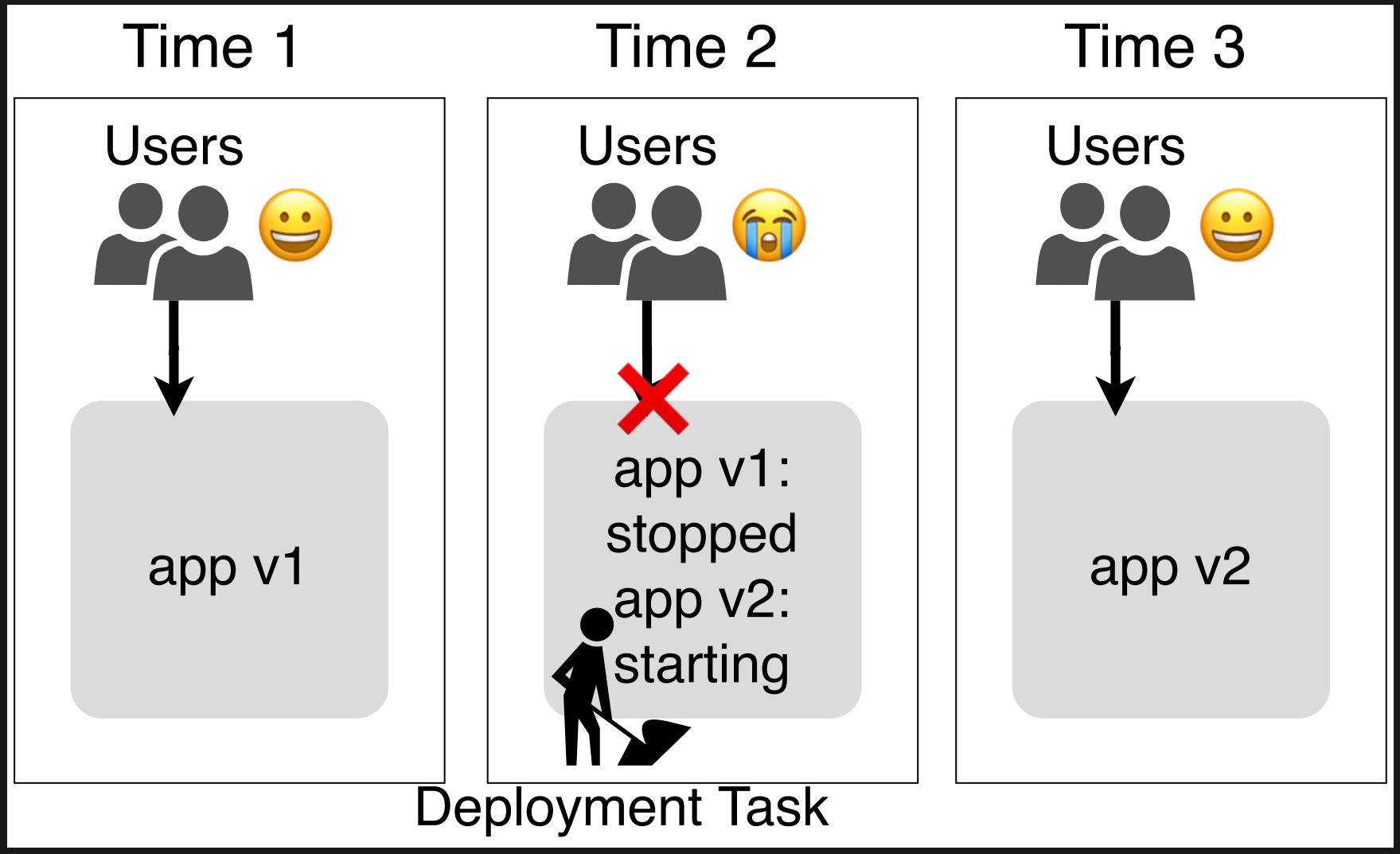
Can Include:

- environment creation
- infrastructure preparations
- configuration injection
- launch of a release/artifact
- new instance health checks
- directing traffic to the release/artifact
- rollback to previous release/artifact

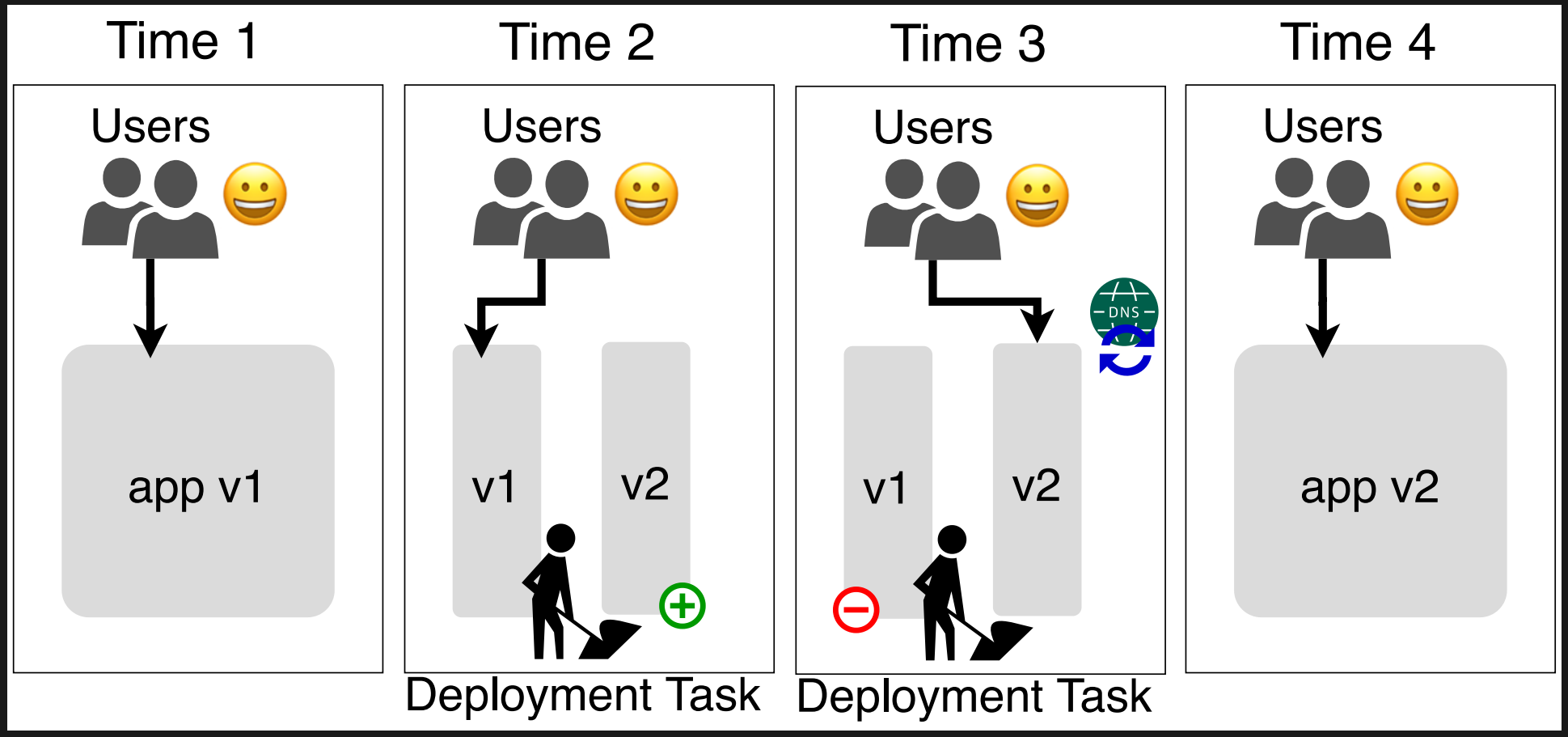
Ship Step - Directing Traffic

There are a lot of ways to do this.

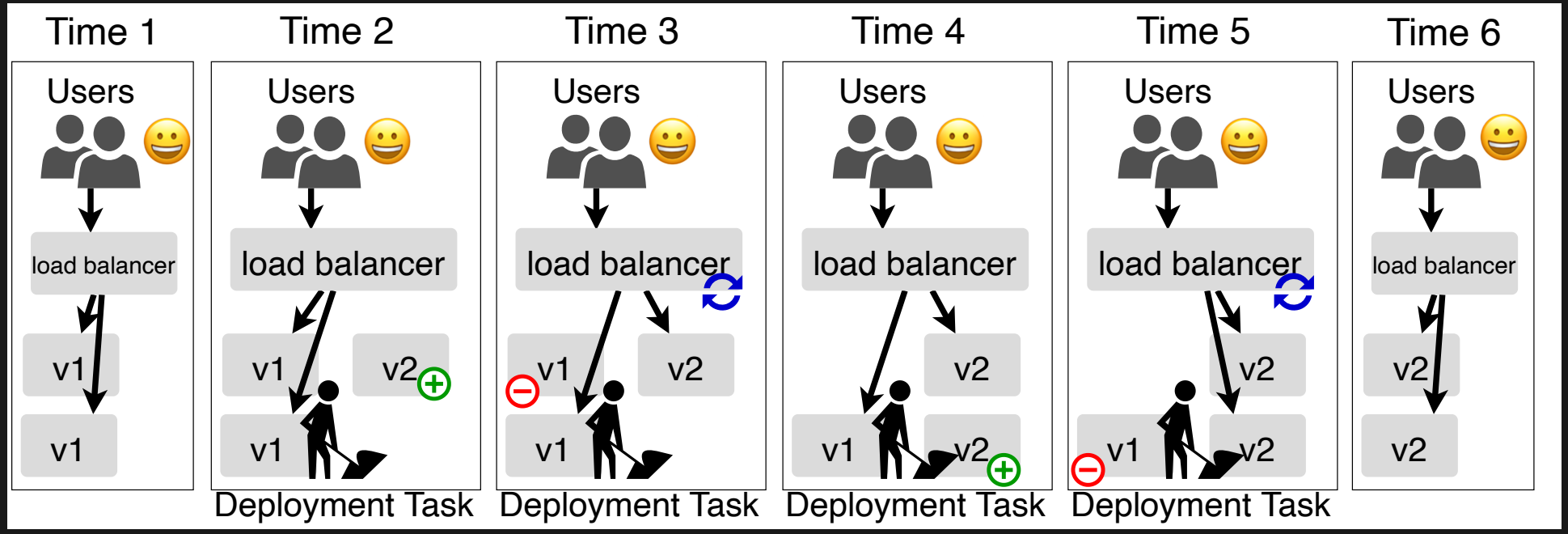
Directing Traffic - Live server



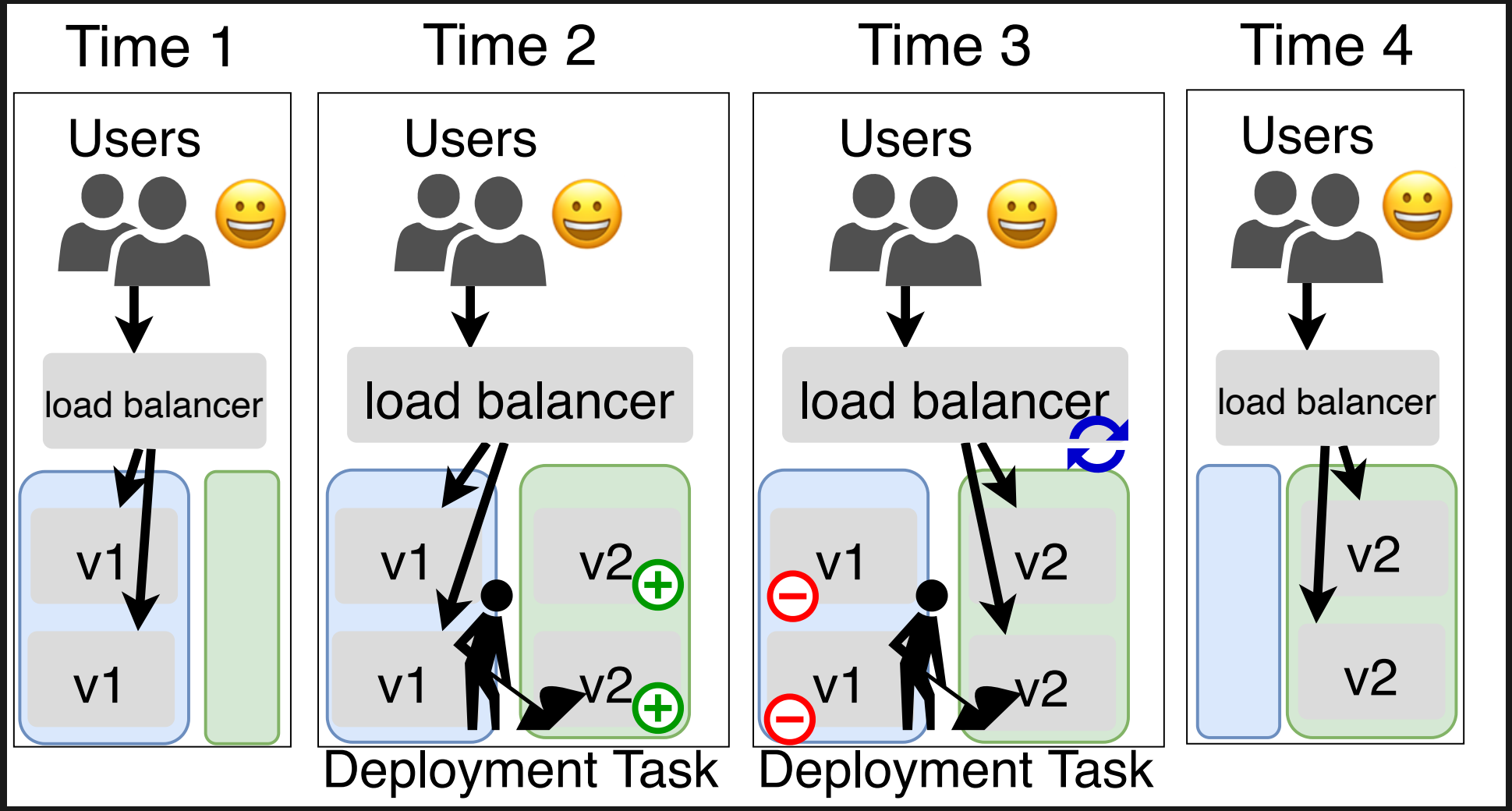
Directing Traffic - IP/DNS swap



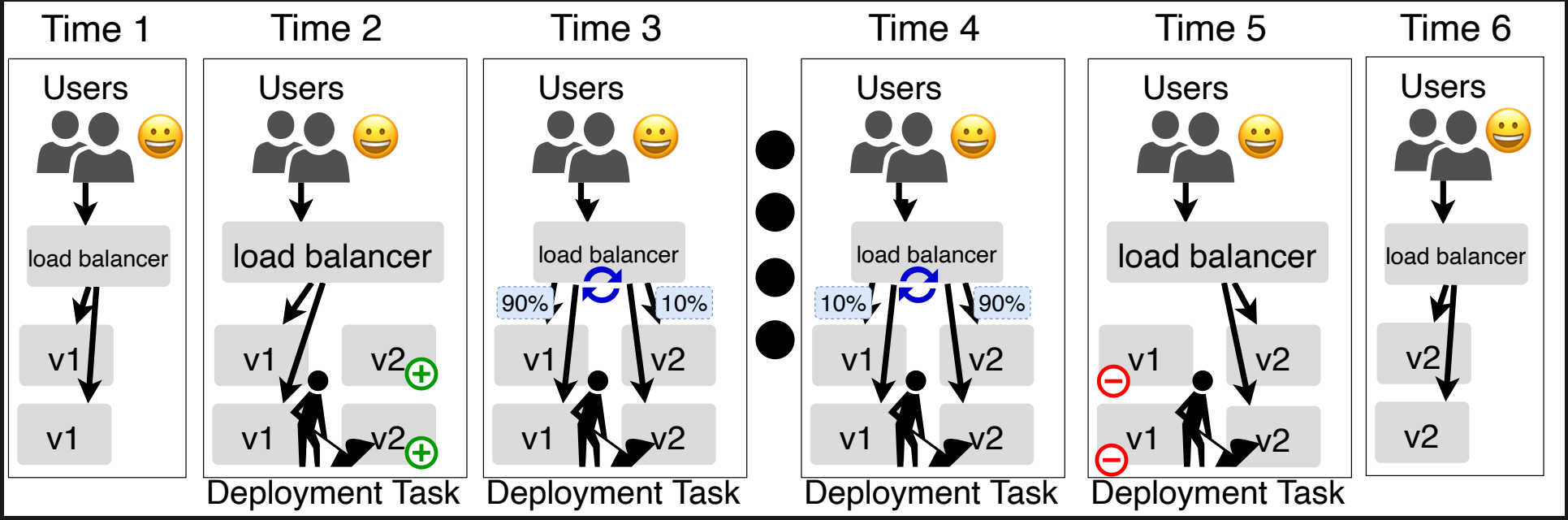
Directing Traffic - Rolling Update



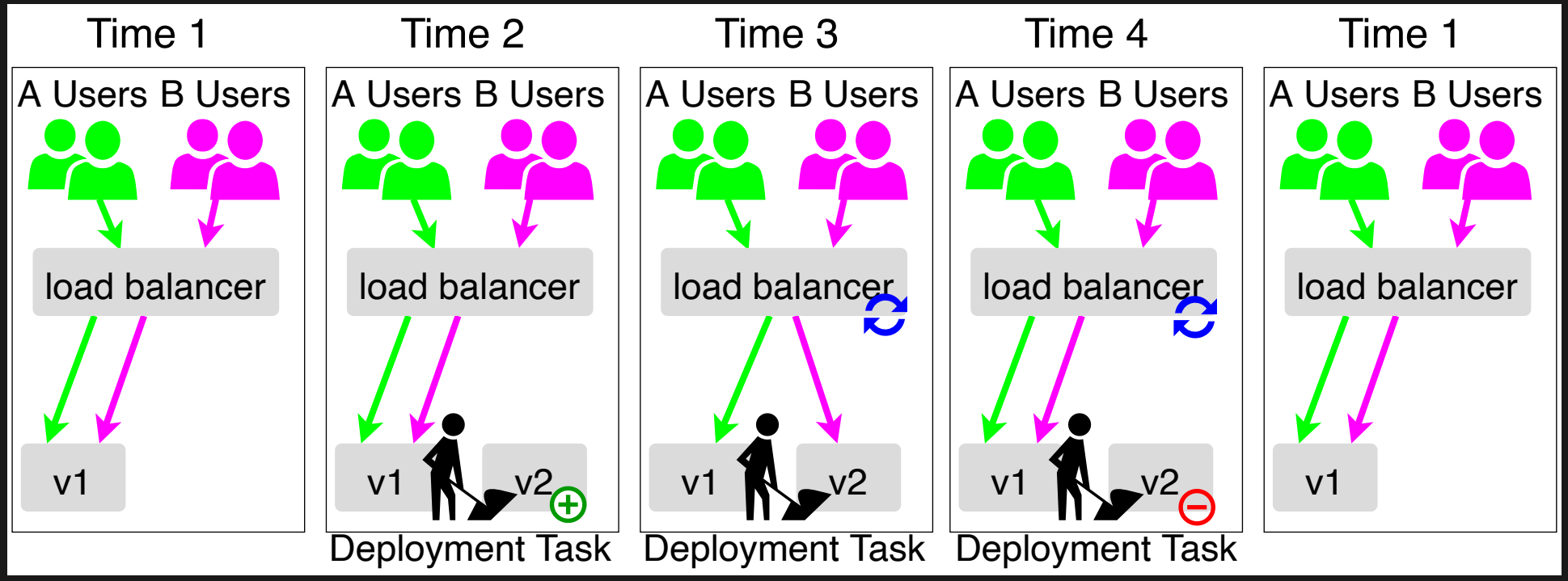
Directing Traffic - Blue/Green



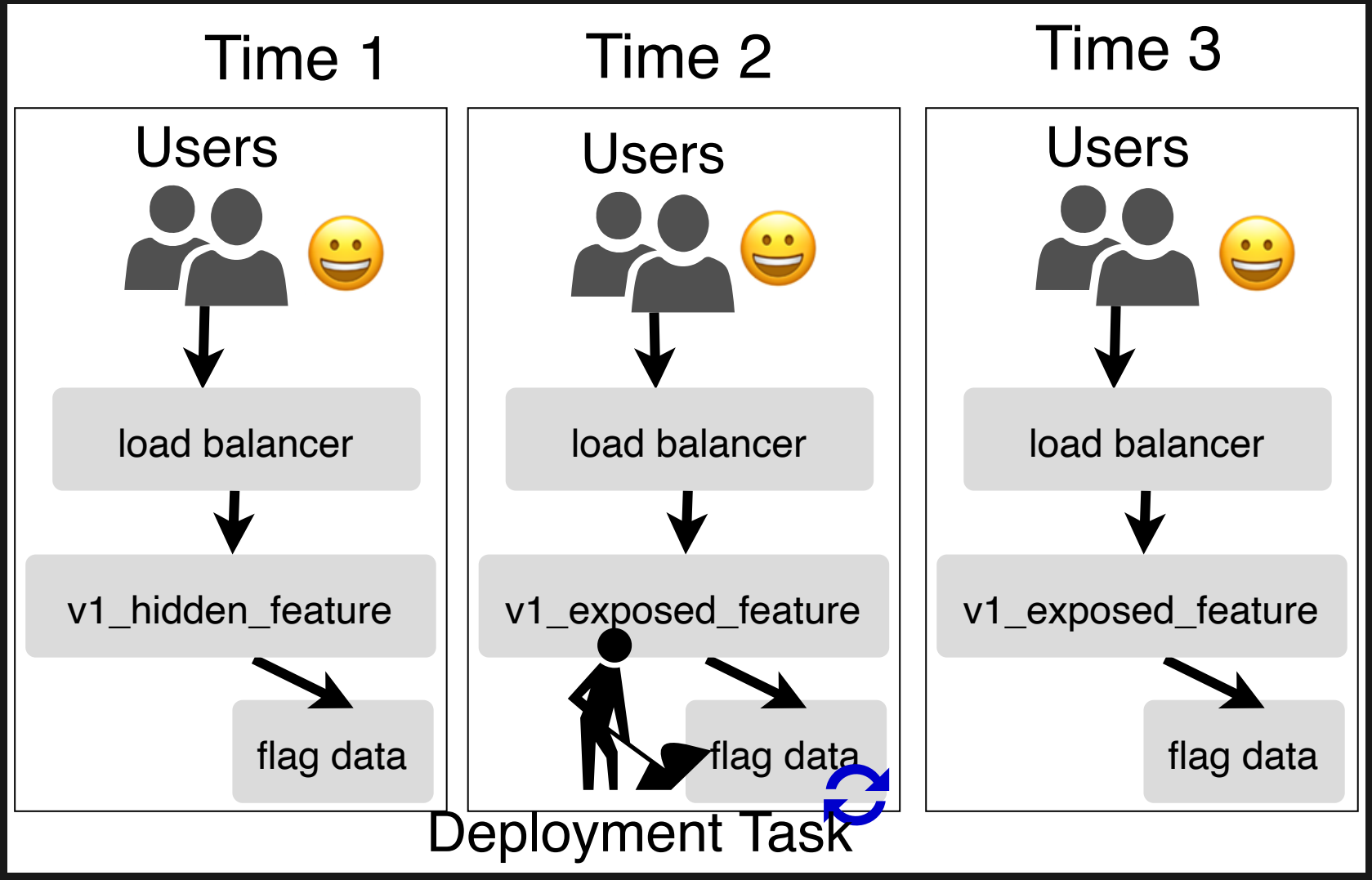
Directing Traffic - Canary



Directing Traffic - A/B Testing



Directing Traffic - Feature Flags



Agenda

- ✅ Preliminaries
- ✅ A General Model
- ➡️ Catalog of Practices
  - ✅ Build
  - ✅ Ship
  - ➡️ Evaluate
- Example Workflows
- Considerations, Caveats, and Advice

The Evaluate Step

Can execute from:

- a CI server
- in the brains of stakeholders
- or other external systems

Evaluate Step Practices

- execute integration or functional tests
- run security assessments
- seek user feedback
- collect availability and performance metrics
- track Key Performance Indicators
- communicate what was learned

Evaluate Step - Communication

Information may be relevant to the next:

- deployment workflow step
- development iteration
- operations/infrastructure change cycle
- organizational strategic plan

Agenda

- ✅ Preliminaries
- ✅ A General Model
- ✅ Catalog of Practices
- ➡️ Example Workflows
- Considerations, Caveats, and Advice

Microservices Entrant

Startup using a popular container platform with hopes of fame and fortune

```
Deployment in_development:
  Build:
    - lint and syntax check locally with IDE
    - unit tests locally with IDE for test driven developm
    - docker build
  Ship:
    - docker-compose up, with the "live server" pattern
  Evaluate:
    - subset of integration tests with mocked api calls
    - feedback from testing determines Go/NoGo for Deploym
test_env

Deployment test_env:
  Build:
```

Weekend Warrior

Uses a Platform as a Service to host hobby/side project

```
Deployment in_development:
  Build:
    - lint and syntax check locally with IDE
    - docker build
  Ship:
    - docker-compose up, with the "live server" pattern
  Evaluate:
    - manual testing with local address:port
    - feedback from testing determines Go/NoGo for Deploym
in_PaaS

Deployment in_PaaS:
  Build:
    - git commit
```

Unseen University

Has multiple sites that use a content management system requiring database connections, file storage, and custom complation

```
Deployment in_development:
  Build:
    - lint and syntax check locally with IDE
    - VMs provided with repo checkout via vagrant shared f
  Ship:
    - vagrant up, creates a disposable environment
    - vagrant kicks off saltstack configuration management
initializes database, loadbalancer, executes the CMS
configuration and compilation script, and starts the servi
  Evaluate:
    - manual testing with VM address:port
    - feedback from testing determines Go/NoGo for Deploym
datacenter_dev_env
```

Agenda

- ✅ Preliminaries
- ✅ A General Model
- ✅ Catalog of Practices
- ✅ Example Workflows
- ➡️ Considerations, Caveats, and Advice

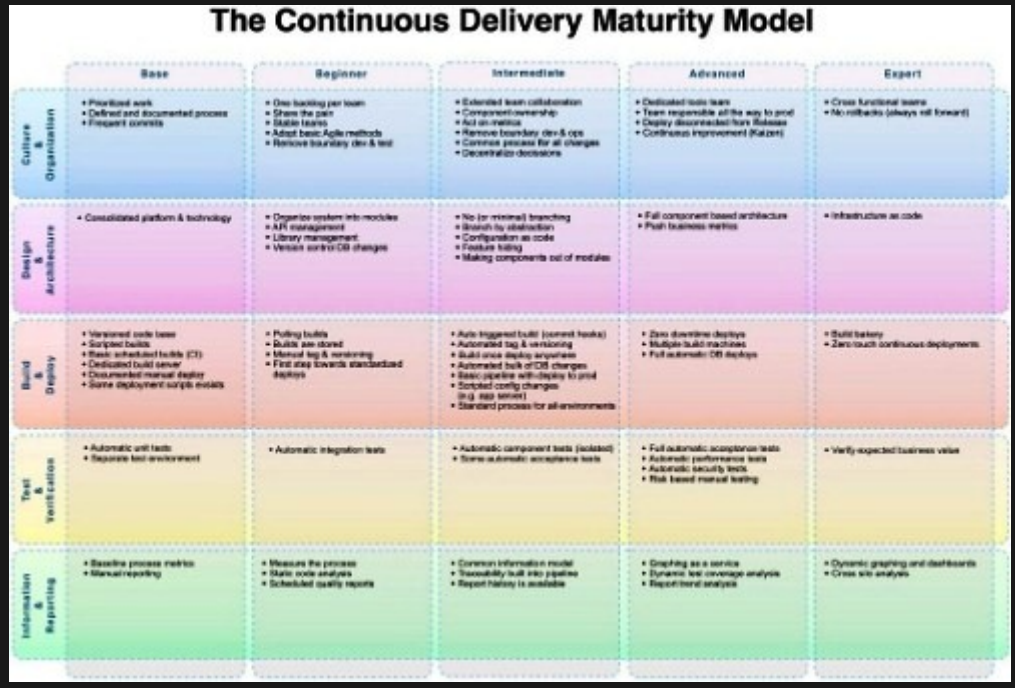
Getting Value from Deployment

A deployment workflow can deliver:

- speed
- safety
- ease of use
- security assurances
- reduce risk
- revert capability
- auditability
- consistency

But, it takes time, effort, and constant vigilance

Maturity Models



Are not qualified to judge you

Assess Your Needs

- Who are you?
- What does your application mean to you?
- What capabilities are practical to implement?

Don't forget

- avoid stale data
- accomodate cache warmup times
- look for partially or undocumented dependencies
- don't make incompatible database changes
- prevent cache poisioning
- monitor for runaway processes
- get stakeholder buy-in

Also

I believe in you!

- get started with the easy stuff
- notice repetitive tasks and reoccurring issues
- you can stop when you have what you need
- use, share, and improve community tools

Agenda

All done!

- ✅ Preliminaries
- ✅ A General Model
- ✅ Catalog of Practices
- ✅ Example Workflows
- ✅ Considerations, Caveats, and Advice

Resources

- Full Stack Python - Deployment: https://www.fullstackpython.com/deployment.html
- Systems development life cycle: https://en.wikipedia.org/wiki/Systems_development_life_cycle
- Release management: https://en.wikipedia.org/wiki/Release_management
- Six Strategies for Application Deployment: https://thenewstack.io/deployment-strategies/
- Semantic Versioning: https://semver.org/
- The Visible Ops Handbook: https://itpi.org/the-visible-ops-book-series/visible-ops-handbook-review/
- The Practice of Cloud System Administration: DevOps and SRE Practices for Web Services: https://everythingsysadmin.com/books.html
- Mykel Alvis - Developing for Deterministic Deliveries, the live version: https://www.youtube.com/watch?v=gTa1fJuPP0E&start=616
- Ansible - Playbook Example: Continuous Delivery and Rolling Upgrades: https://docs.ansible.com/ansible/latest/user_guide/guide_rolling_upgrade.html
  System Deployment Tips and Techniques httphttps://www.ambysoft.com/essays/deploymentTips.html
- USGS - Deployment Best Practices: https://www.usgs.gov/products/software/software-management/deployment-best-practices
- Kubernetes for Sysadmins – Kelsey Hightower at PuppetConf 2016: https://www.youtube.com/watch?v=HlAXp0-M6SY
- Continuous Integration & Delivery (CI/CD) for Kubernetes Using CircleCI & Helm: https://medium.com/velotio-perspectives/continuous-integration-delivery-ci-cd-for-kubernetes-using-circleci-helm-b8b0a91ef1a3
- Feature Toggles (aka Feature Flags): https://martinfowler.com/articles/feature-toggles.html
- Matching Supply with Demand: An Introduction to Operations Management http://cachon-terwiesch.net/3e/
- Principles behind the Agile Manifesto: https://agilemanifesto.org/principles.html
- The Twelve-Factor App: https://12factor.net/
- Cover image: https://www.usgs.gov/media/images/cartographers-field
- Maturity Model: https://www.infoq.com/articles/Continuous-Delivery-Maturity-Model/