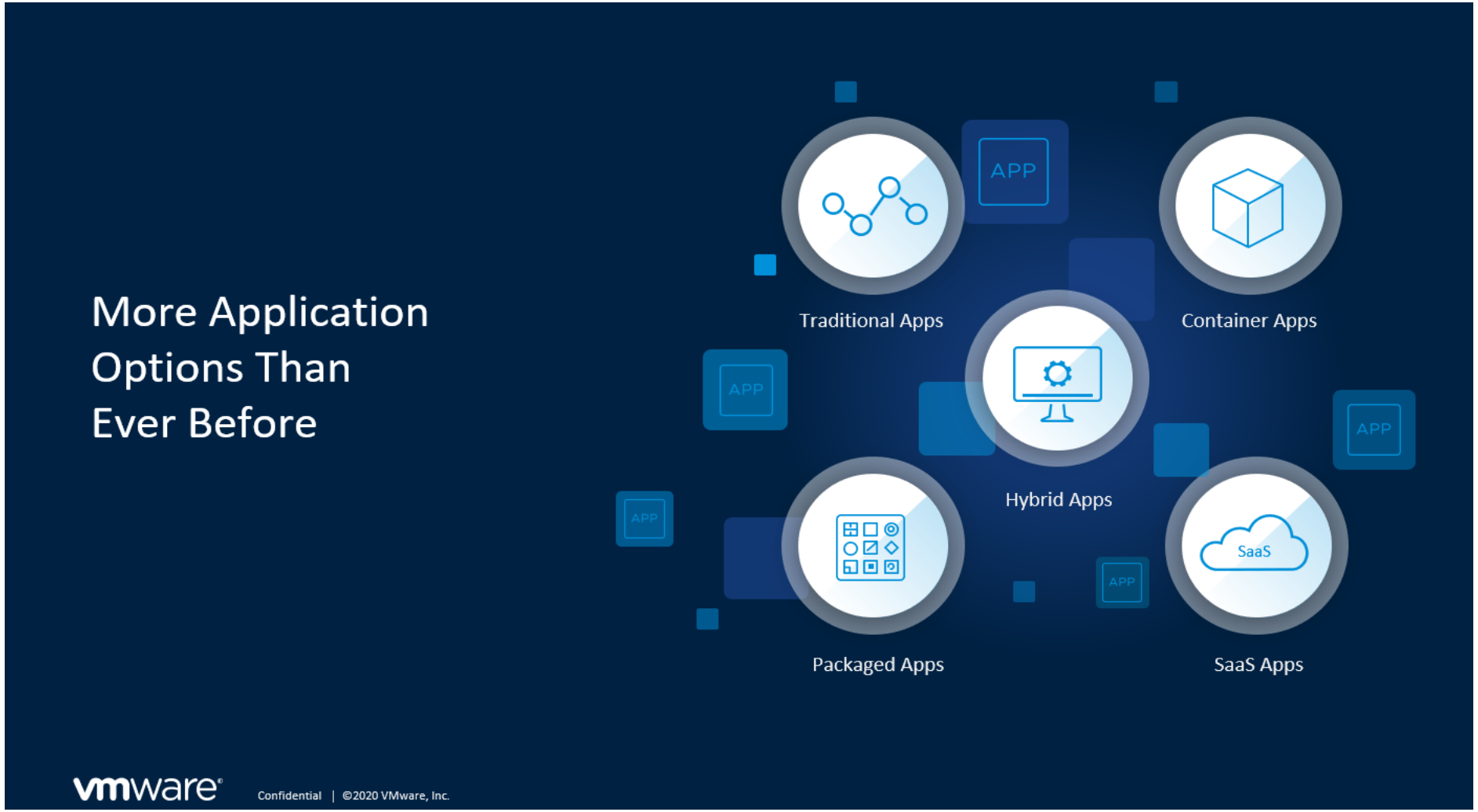
VMWare Launch - Working with Cloud

Application Options – 12 Factor App

Allen R. Sanders
Senior Technology Instructor

Class Roadmap



The Twelve-Factor App

Develop Intelligence

- Methodology for building SaaS applications for the Cloud
- Uses a ***declarative*** (vs. ***imperative***) format for defining setup automation
- Maintains clean contract with the underlying operating system
- Drives toward consistent interface but pluggable implementation based on target OS / platform (see SOLID principles)

The Twelve-Factor App

- Supports modern cloud platforms
- Targets minimal divergence between development and production to enable continuous deployment (agility)
- Targets “elastic scalability” – the ability to dynamically automate scaling of a system to quickly adjust to demand while optimizing cost

I. Codebase

Develop Intelligence

One codebase tracked in revision control, many deploys

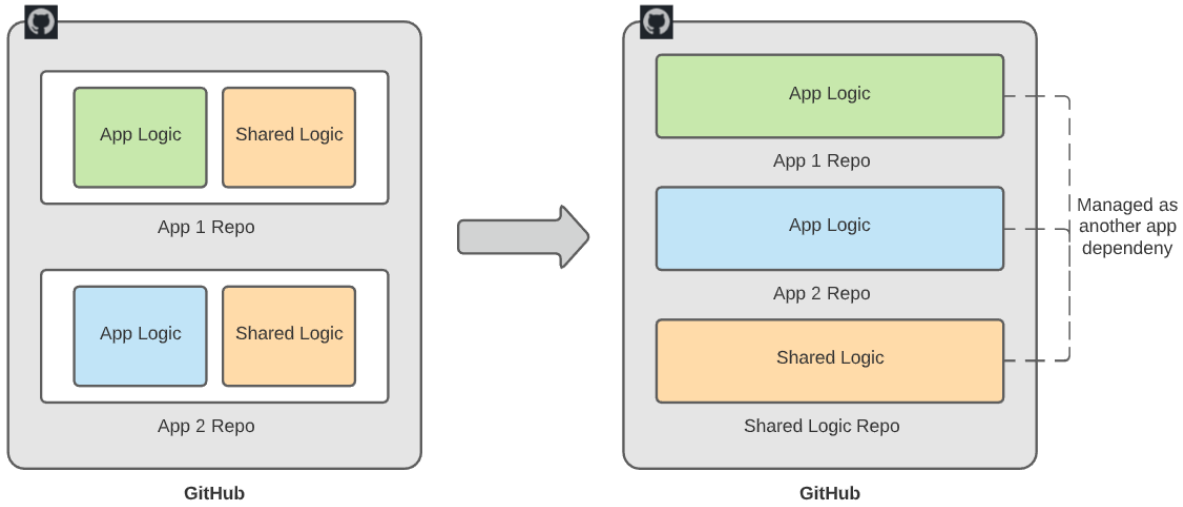
- A twelve-factor app is always tracked in a version control system in a ***code repo***
- Target is one-to-one mapping between codebase and app
- Multiple apps repeating the same code is considered a violation – the shared code should be factored into its own repo

I. Codebase

Develop Intelligence

One codebase tracked in revision control, many deploys

- There is only one codebase but there can be multiple deployments across multiple environments (e.g. development, QA, staging, production)
- Codebase is the same across all deployments, but environments may have different versions (code branches) and different configuration



II. Dependencies

Develop Intelligence

Explicitly declare and isolate dependencies

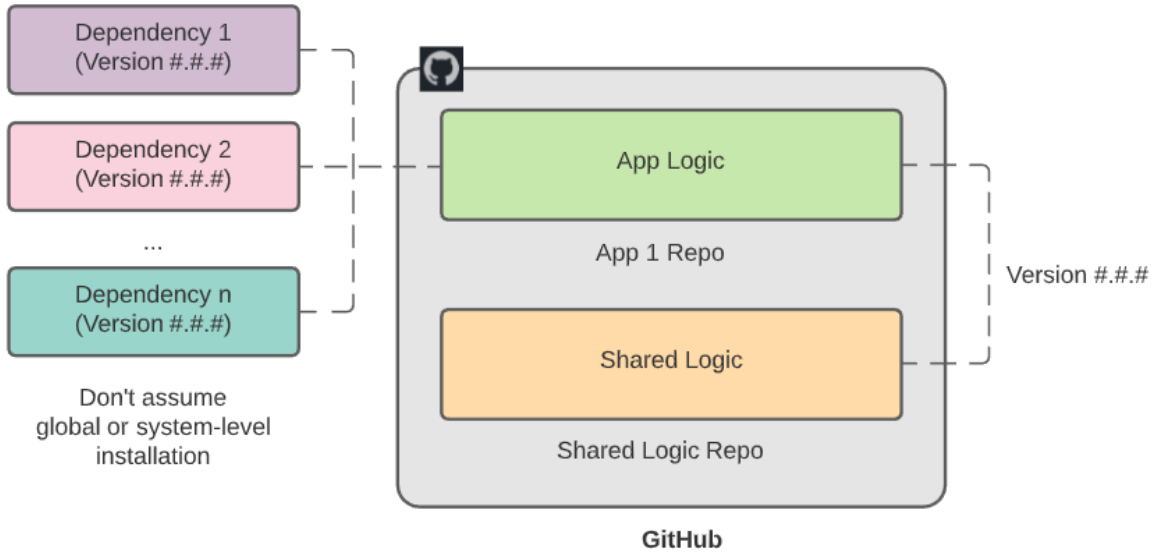
- Depending on the platform, dependencies can be globally-defined or locally-defined (i.e. local to an application implementation)
- The twelve-factor app explicitly declares all dependencies (name and version) locally using a dependency declaration manifest
- The twelve-factor app also leverages dependency isolation to ensure local dependencies do not get overridden by a different scope

II. Dependencies

Develop Intelligence

Explicitly declare and isolate dependencies

- Does not rely on implicit existence of system tools – app is a complete bundle
- Helps simplify onboarding of new developer (no assumptions required)



III. Config

Develop Intelligence

Store config in the environment

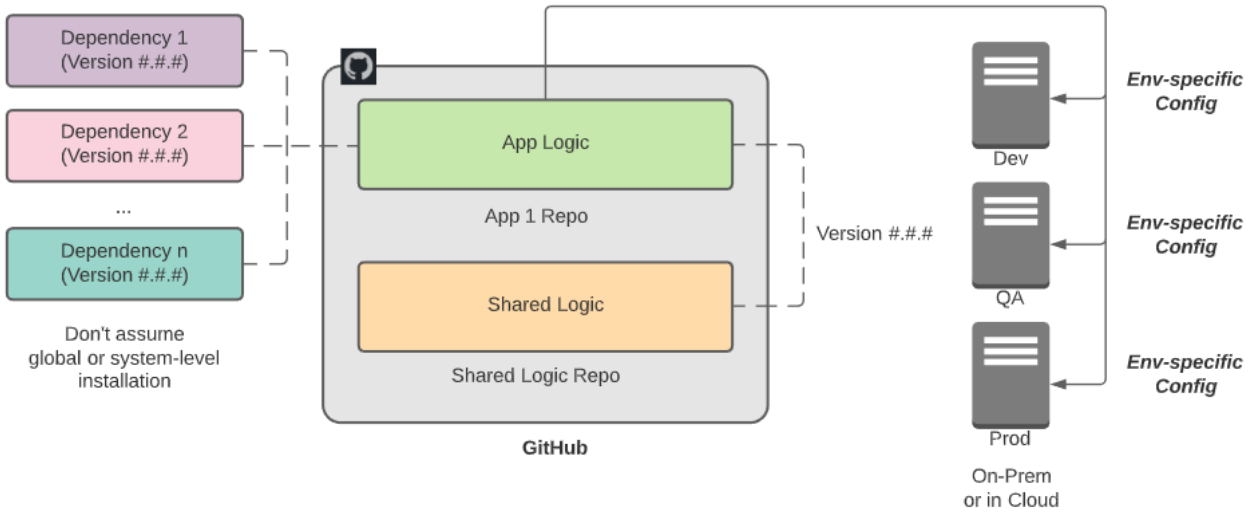
- Configuration data is usually settings that will vary from environment-to-environment (e.g. connection strings, target URL's, etc.)
- The twelve-factor app requires that this information be separate from the code (cannot use constants in the code to define)

III. Config

Develop Intelligence

Store config in the environment

- The twelve-factor app stores this information in environment variables (variables / values specific to the target machine and its environment)
- Helps to minimize “leakage” of sensitive configuration data either checked into the repo as code constants or config files



IV. Backing Services

Treat backing services as attached resources

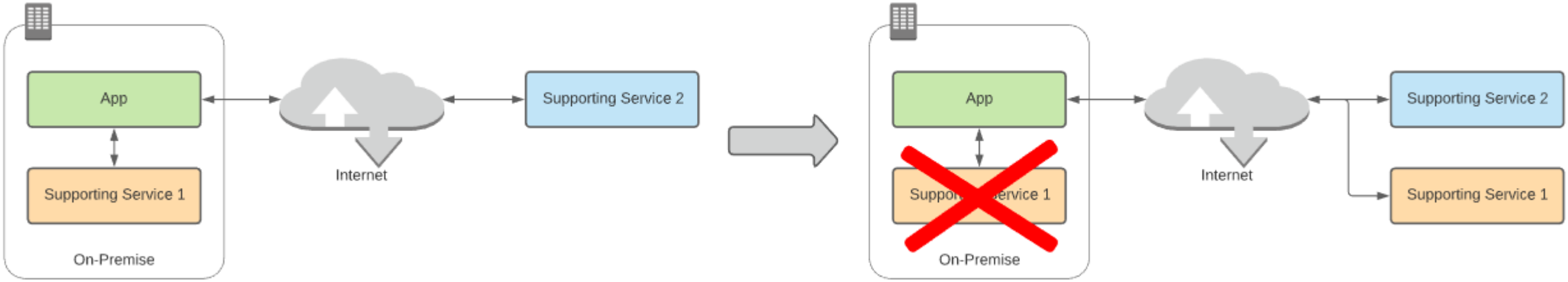
- A backing service is a service that the application consumes over the network or Internet (e.g. database, e-mail services, etc.)
- These backing services are sometimes supported by third parties
- The twelve-factor app does not make a distinction between local and third-party services – all are considered attached resources

IV. Backing Services

Develop Intelligence

Treat backing services as attached resources

- By avoiding this distinction, the twelve-factor app can practice looser-coupling with its dependencies regardless of locality
- Enables easier swapping between dependencies to ensure optimal target resource is employed



V. Build, Release, Run

Develop Intelligence

Strictly separate build and run stages

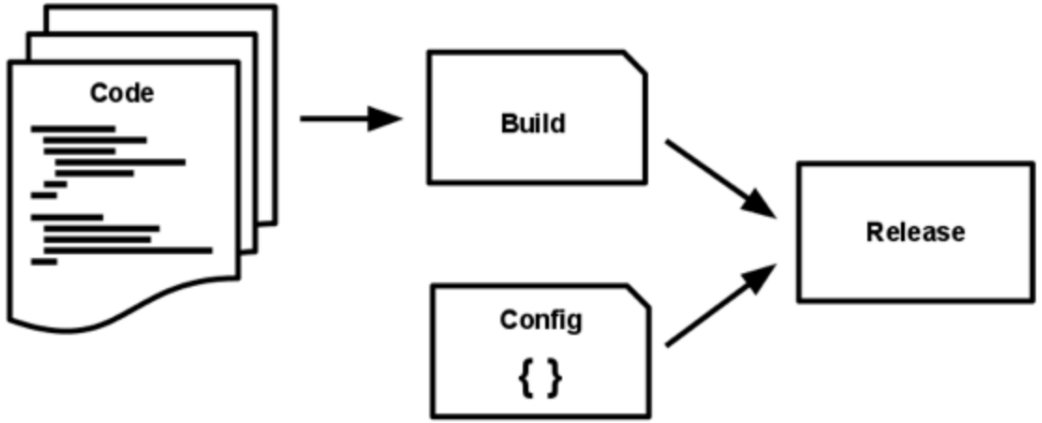
- The twelve-factor app uses strict separation between the build, release and run stages
- Changes will not be made to the release or run stage independent from the build stage

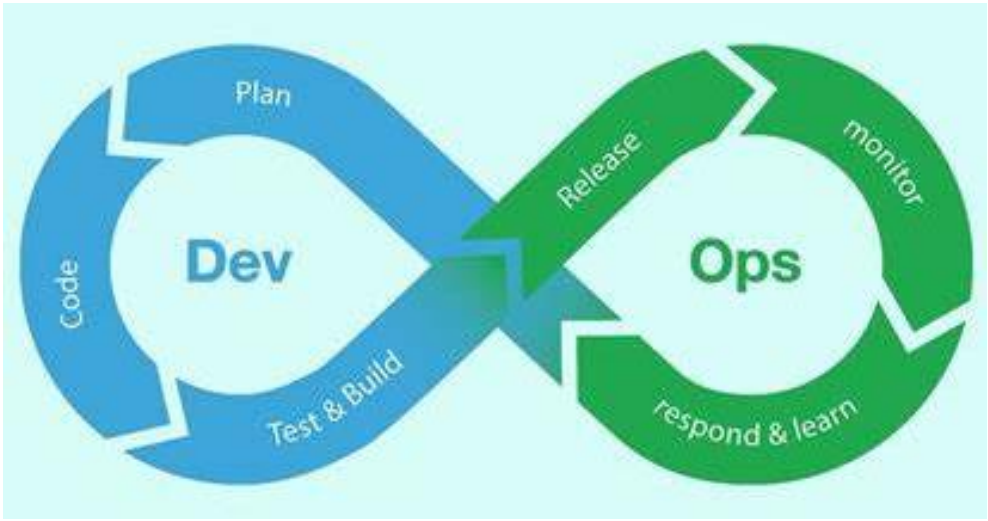
V. Build, Release, Run

Develop Intelligence

Strictly separate build and run stages

- Helps maintain an orderly (and repeatable) unidirectional flow through the stages
- Also, allows for leverage of a unique identifier for each end-to-end flow through the stages for traceability and troubleshooting





VI. Processes

Execute the app as one or more stateless processes

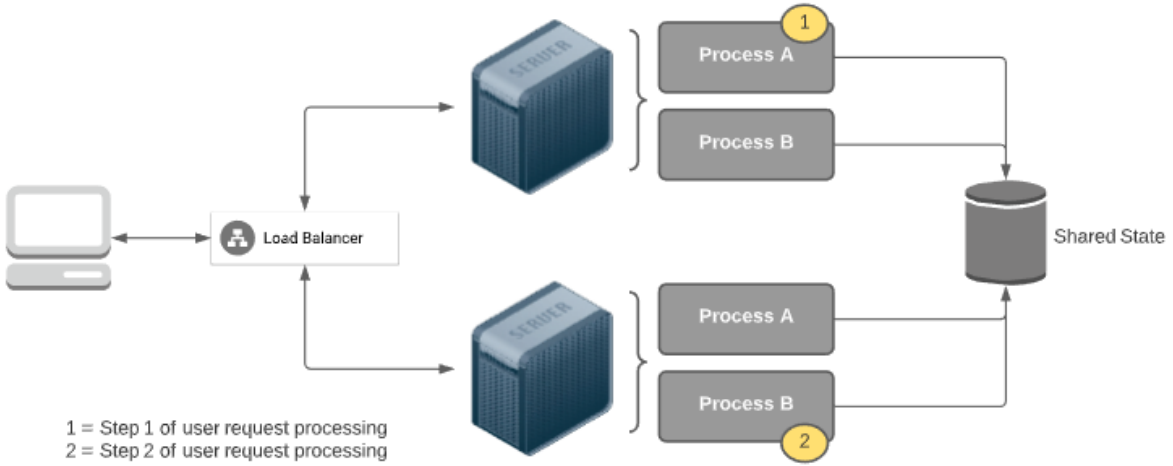
- The twelve-factor app does not rely on the sharing of state across processes through direct transfer of information
- Instead, shared state is stored via a separate backing service external to a given process but accessible from multiple

VI. Processes

Develop Intelligence

Execute the app as one or more stateless processes

- Otherwise, you create tight coupling between the processes that make up a complex system
- Also, you limit scalability because shared state across processes requires “stickiness” for an end-to-end request (if not externally managed)



VII. Port Binding

Develop Intelligence

Export services via port binding

- Sometimes apps rely on a separate component or container for execution environment (e.g. web app relies on an external webserver container)
- The twelve-factor app does not rely on runtime injection of an external container, but instead is completely self-contained

VII. Port Binding

Develop Intelligence

Export services via port binding

- Any required container will be captured within and exposed from the app environment via binding to a specific port (or ports) for runtime access
- By keeping the application self-contained, you reduce coupling to other external components and minimize required assumptions about runtime environment

```
`docker run -it --rm -p 8000:5000 -e ASPNETCORE_URLS=http://+:5000 --name demo-container demo-image`
```

VIII. Concurrency

Develop Intelligence

Scale out via the process model

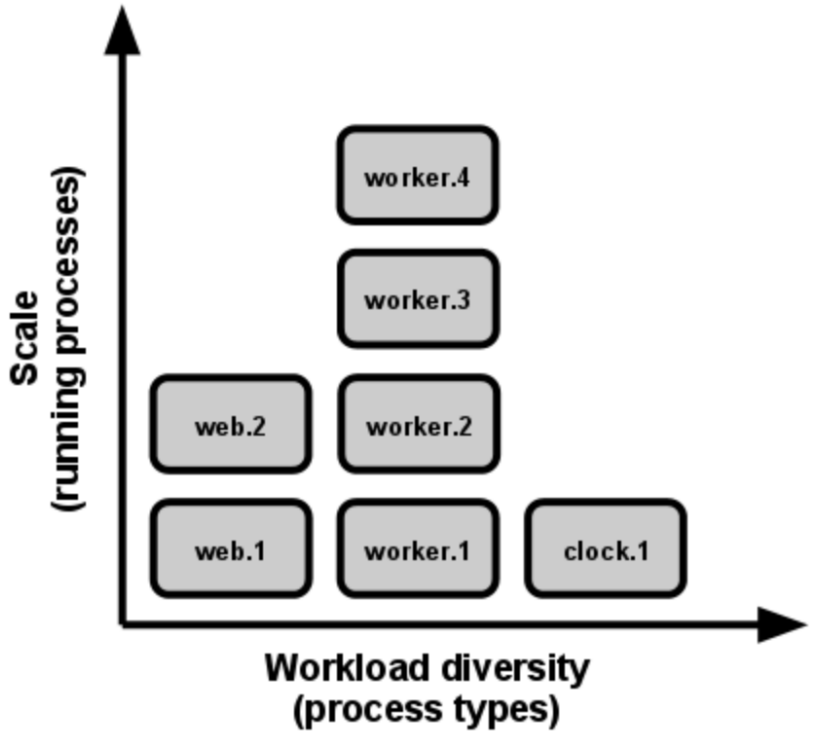
- Processes are a “first class citizen”
- Allows an architect or developer to be intentional about what processes get created and how they are utilized to service a request
- AKA horizontal scaling (vs. vertical scaling)

VIII. Concurrency

Develop Intelligence

Scale out via the process model

- Utilizes the underlying OS to manage the processes but the application maintains visibility
- Enables concurrency – the ability to service multiple types of processing for a given workflow in parallel (scalability and performance)



IX. Disposability

Develop Intelligence

Maximize robustness with fast startup and graceful shutdown

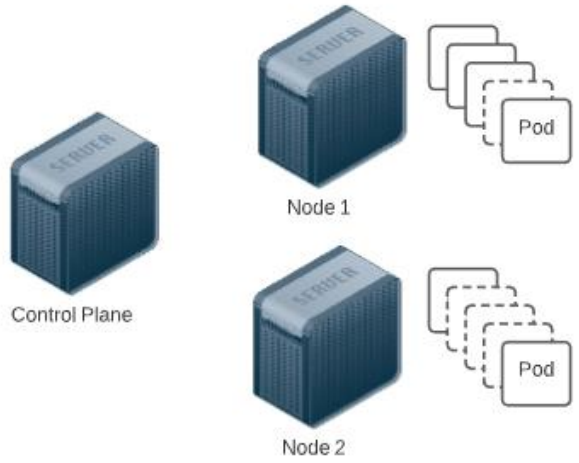
- The twelve-factor app's processes are disposable – they can be started or stopped as required and quickly (when required)
- Enables elastic scalability – quick response to demand (up or down)
- Processes should minimize startup time – whether for single instances or multiple instances of the process (redundancy)

IX. Disposability

Develop Intelligence

Maximize robustness with fast startup and graceful shutdown

- Processes should be able to handle shutdown gracefully – since unneeded processes may need to be destroyed as demand decreases
- Graceful shutdown should include the ability to handle crashes or error conditions



Container Orchestration

X. Dev / Prod Parity

Develop Intelligence

Keep development, staging and production as similar as possible

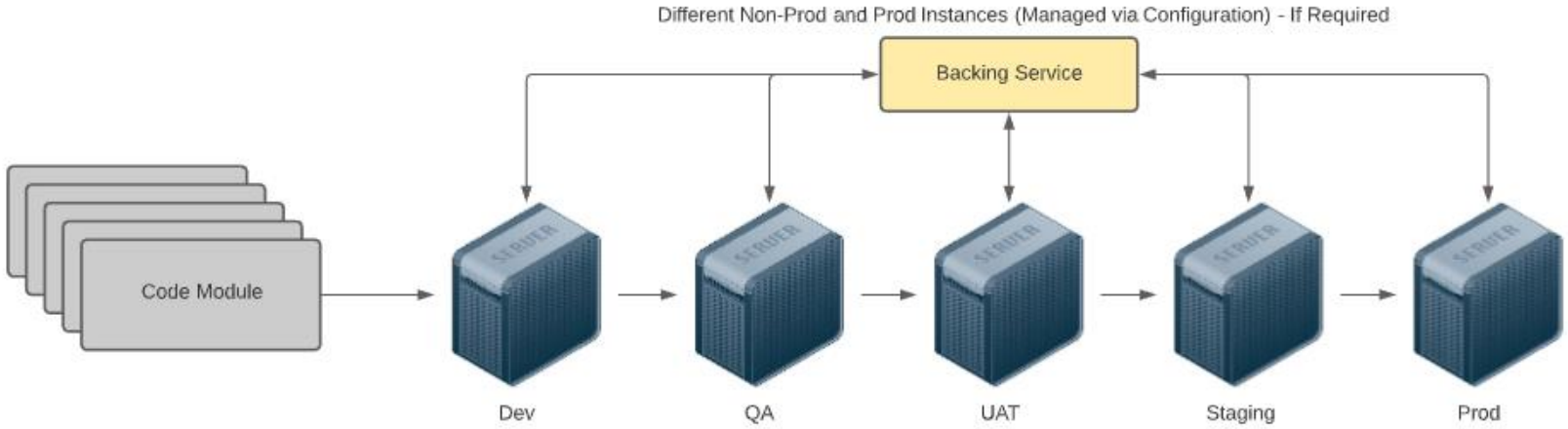
- The twelve-factor app is designed for continuous deployment – multiple pushes, sometimes multiple times a day, in response to ongoing code changes (agility)
- This does not mean that each environment will have the same version or branch deployed

X. Dev / Prod Parity

Develop Intelligence

Keep development, staging and production as similar as possible

- However, modifications to the codebase should be unidirectional to ensure potential impact from change is sufficiently accounted for
- Also, the twelve-factor app targets utilization of the same backing services regardless of environment, targeting variance in configuration alone



XI. Logs

Develop Intelligence

Treat logs as event streams

- Logs from the execution of an app should be streams of aggregated, time-order events collected as output from the running processes
- The twelve-factor app incorporates routing of its corresponding output streams to a separate platform charged with log management (SRP)
- The twelve-factor app relies on another process to aggregate, normalize and scrub the log data into usable information for end-to-end analysis
- Key because there may be relevant logs for a transaction over-and-above the app itself – other components in addition to the app

XII. Admin Processes

Develop Intelligence

Run admin / management tasks as one-off processes

- Admin / management tasks are activities or instructions to be executed outside of the normal application flow (e.g. database migrations, administrative script execution)
- The twelve-factor app treats these tasks as one-off process, ad hoc and separate from the main application (Separation of Concerns)
- These one-off processes should be run in an identical environment against the same codebase and config
- Should leverage the same dependencies (including versions) as are employed with the application itself – difference is in the workflow only

Quick Discussion – Best Practices

Develop Intelligence

Have you observed any similarities between SOLID and 12-factor?

Have you observed differences?

What do you see as the potential benefits of leveraging these best practices in building out systems for the Cloud?

What do you see as the potential challenges?

Break (10 min.)

THANK YOU

Develop
Intelligence


Develop
Intelligence