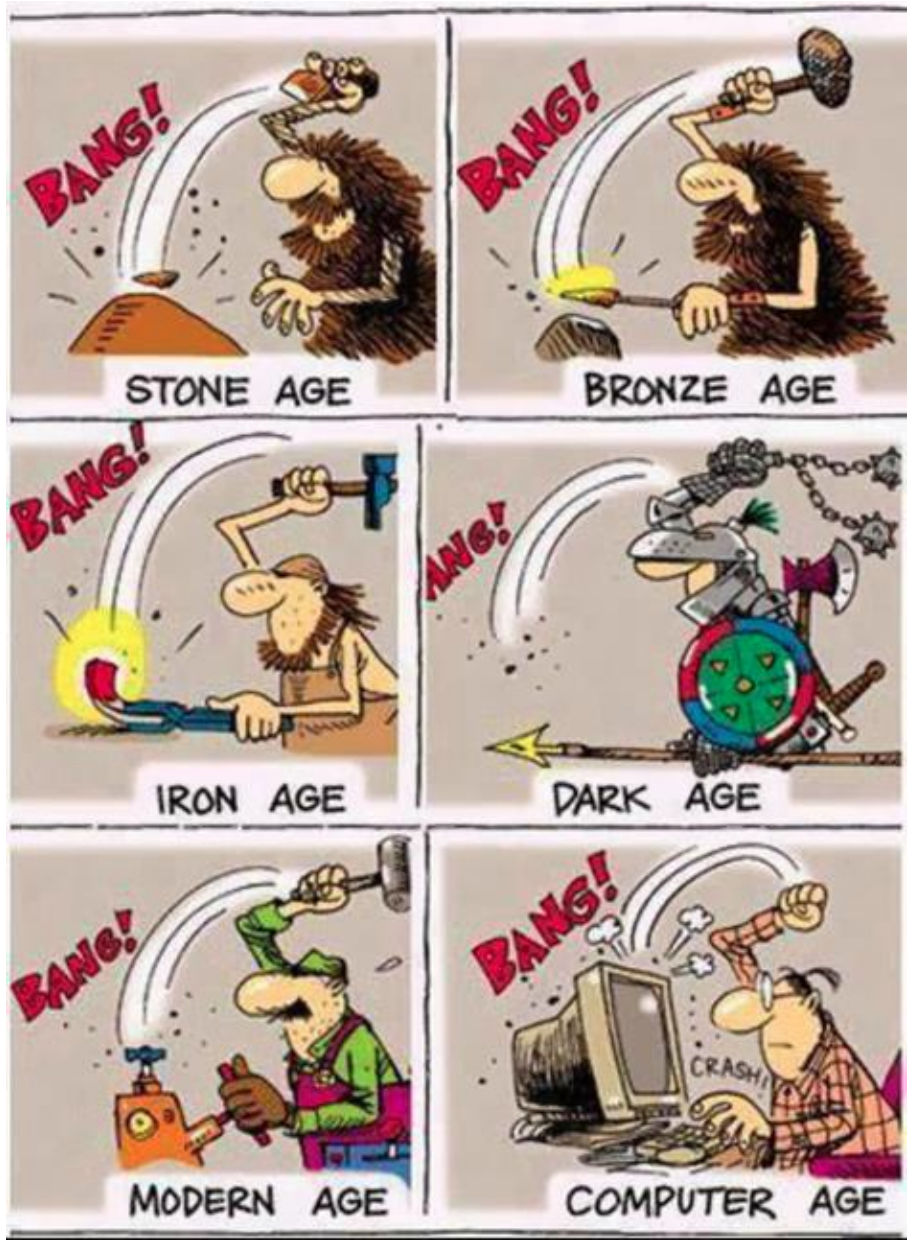
Cloud Native Application Development
The Twelve Factor App Methodology

Robert Baumgartner
Senior Solution Architect

Red Hat

Humans had been problem solvers...

until they've decided to become PROGRAMMERS



Red Hat

The Twelve-Factor App

- **A methodology**
- Manifesto
- Best practices
- Principles

Created by heroku https://12factor.net/

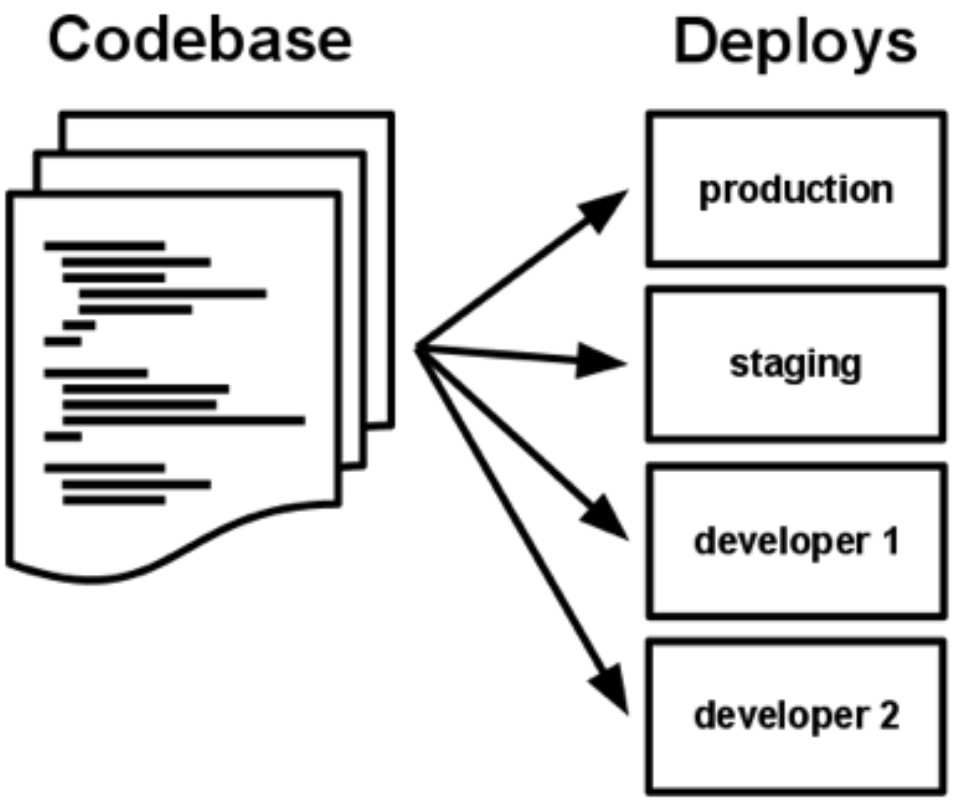
Red Hat

1. Codebase
2. Dependencies
3. Config
4. Backing services
5. Build, release, run
6. Processes
7. Port binding
8. Concurrency
9. Disposability
10. Dev/prod parity
11. Logs
12. Admin processes

Red Hat

1 - Codebase

One codebase tracked in revision control, many deploys

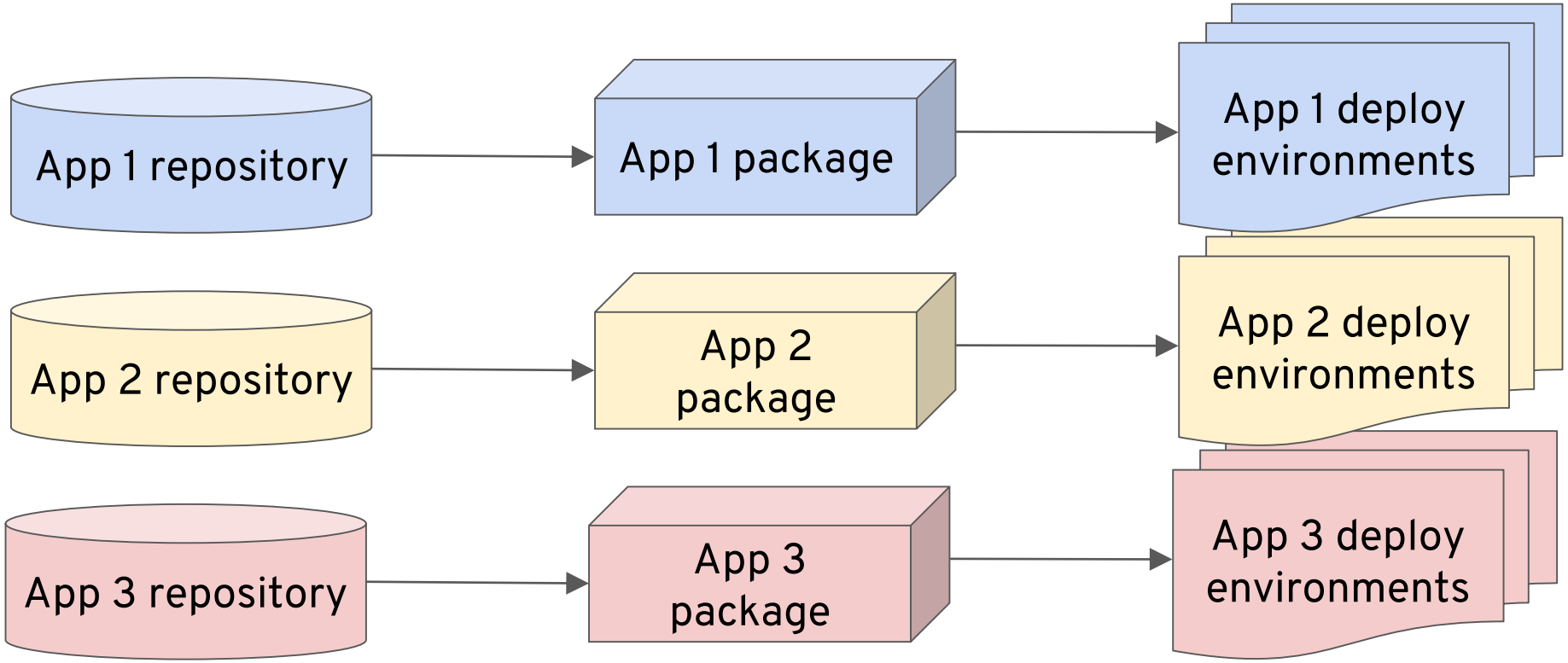


One codebase = one app

Red Hat

1 - **Codebase** (What does it mean?)

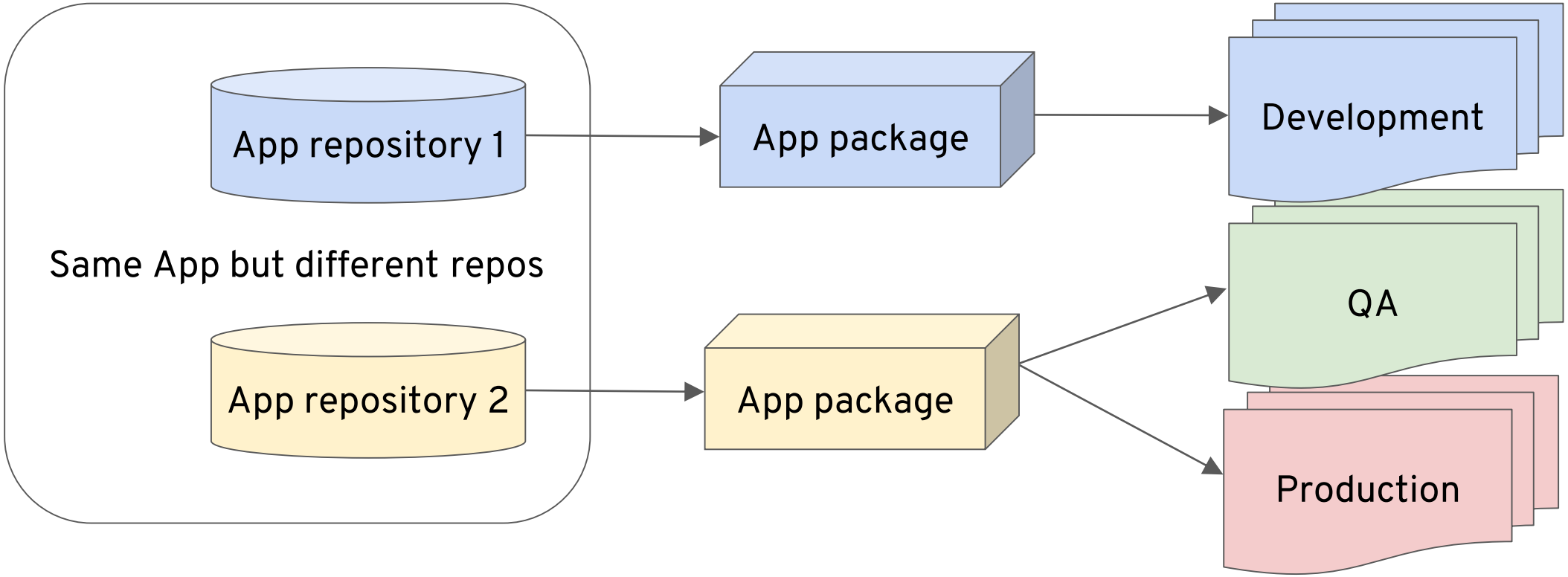
Use Version Control - But use it the right way!



Red Hat

1 - **Codebase** (DO NOTs)

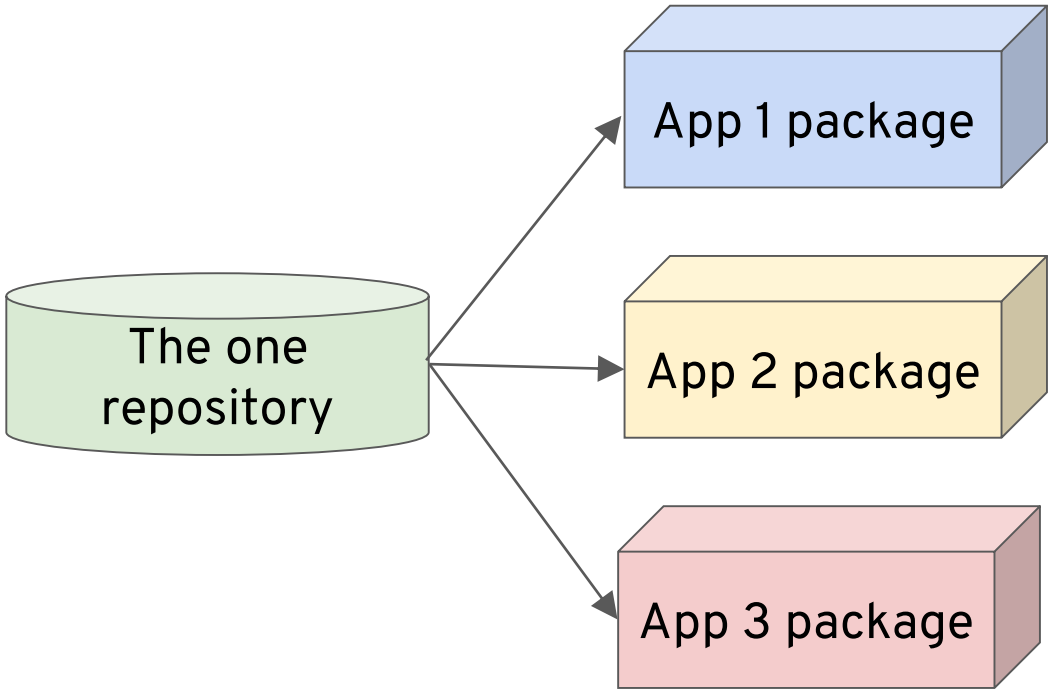
DO NOT have different codebases for different deployments



Red Hat

1 - **Codebase** (DO NOTs)

DO NOT have multiple apps and docs in the same repository

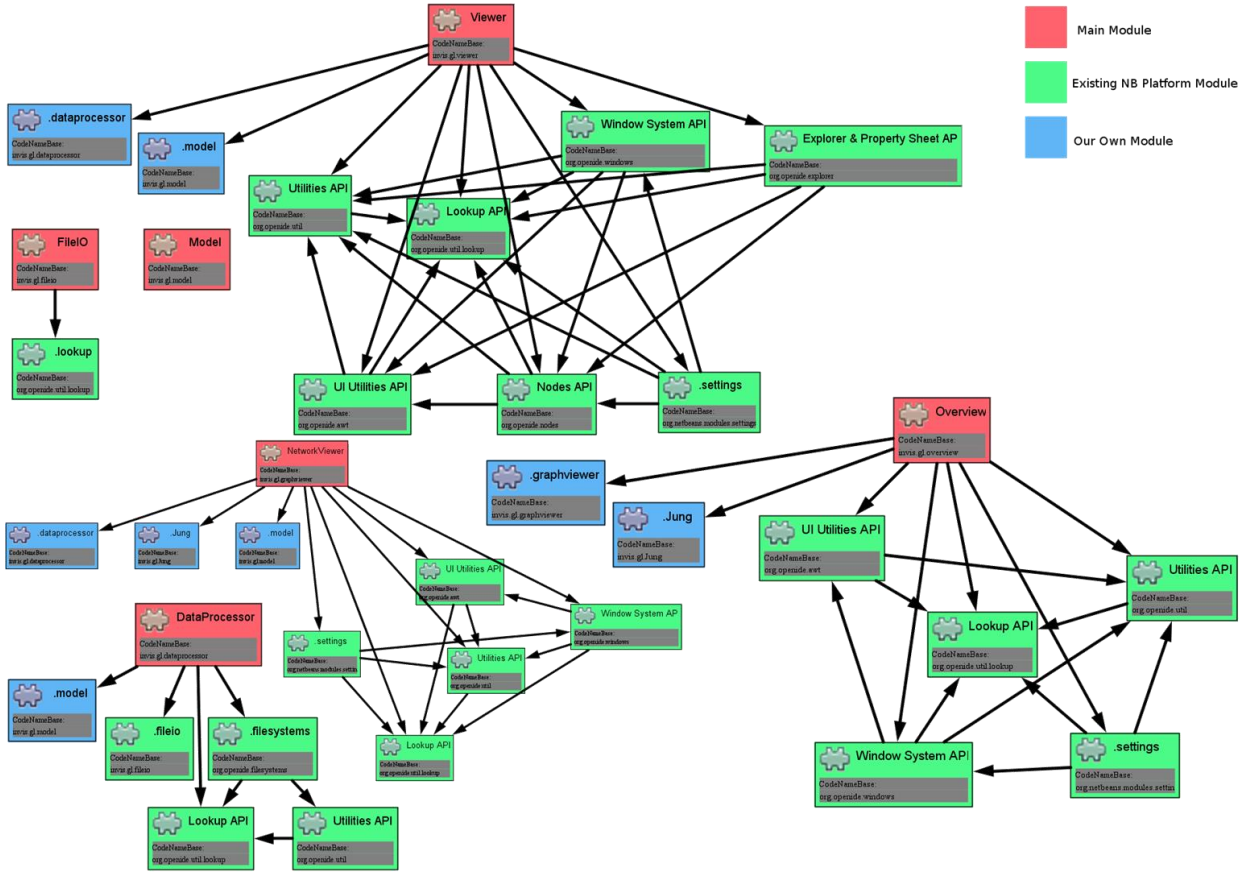


DEMO

Red Hat

2 - Dependencies

Explicitly declare and isolate dependencies

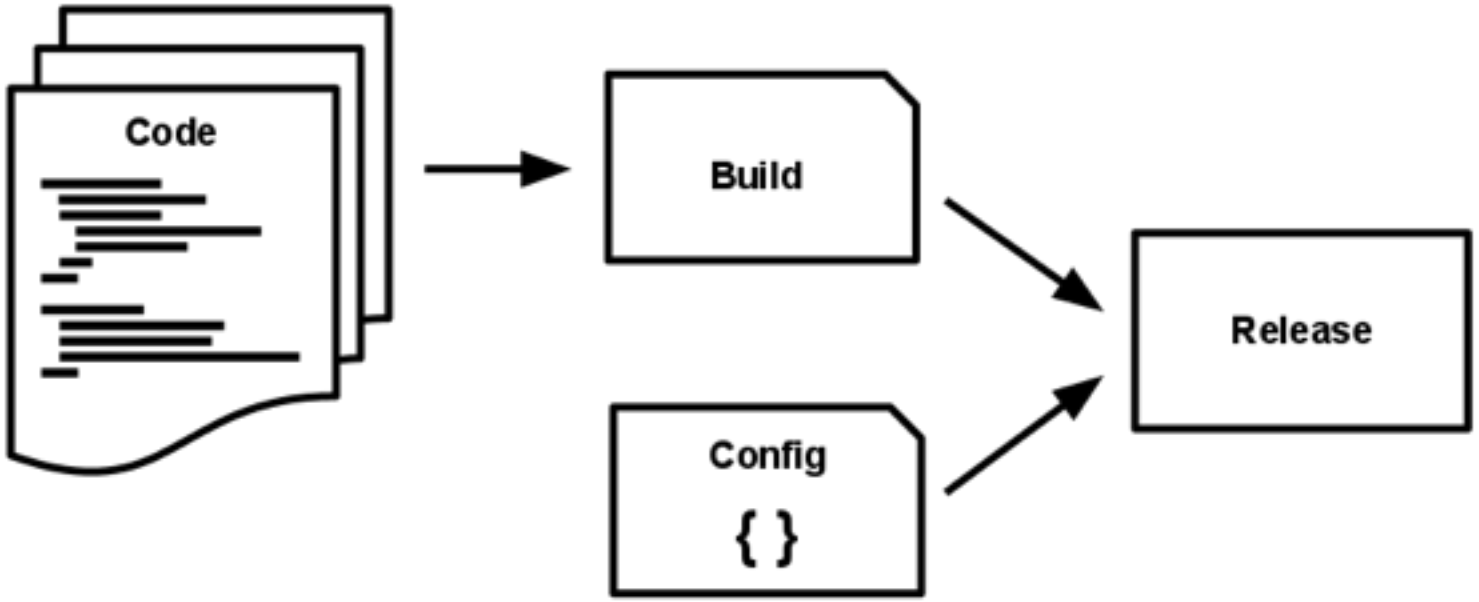


2 - **Dependencies** (What does it mean?)

- Use a package manager to avoid dependency hell.
- Don't commit dependencies in the codebase repository.

5 - Build, release, run

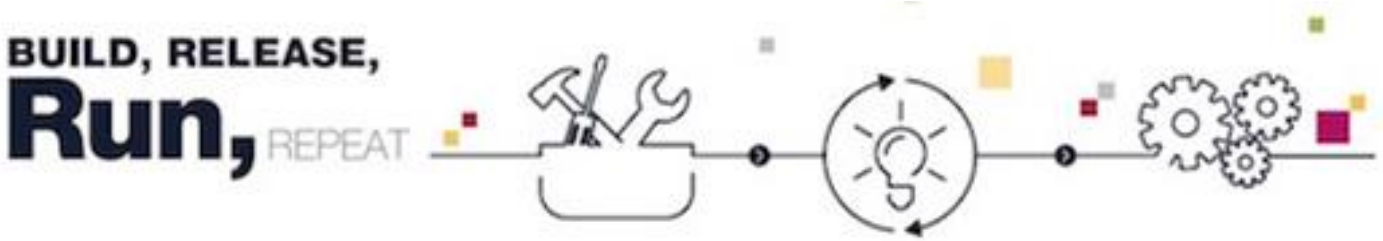
Strictly separate build and run stages



Red Hat

5 - **Build, release, run** (What does it mean?)

Use strict separation between the build, release, and run stages.



- Every release should always have a unique release ID
- Releases should allow rollbacks

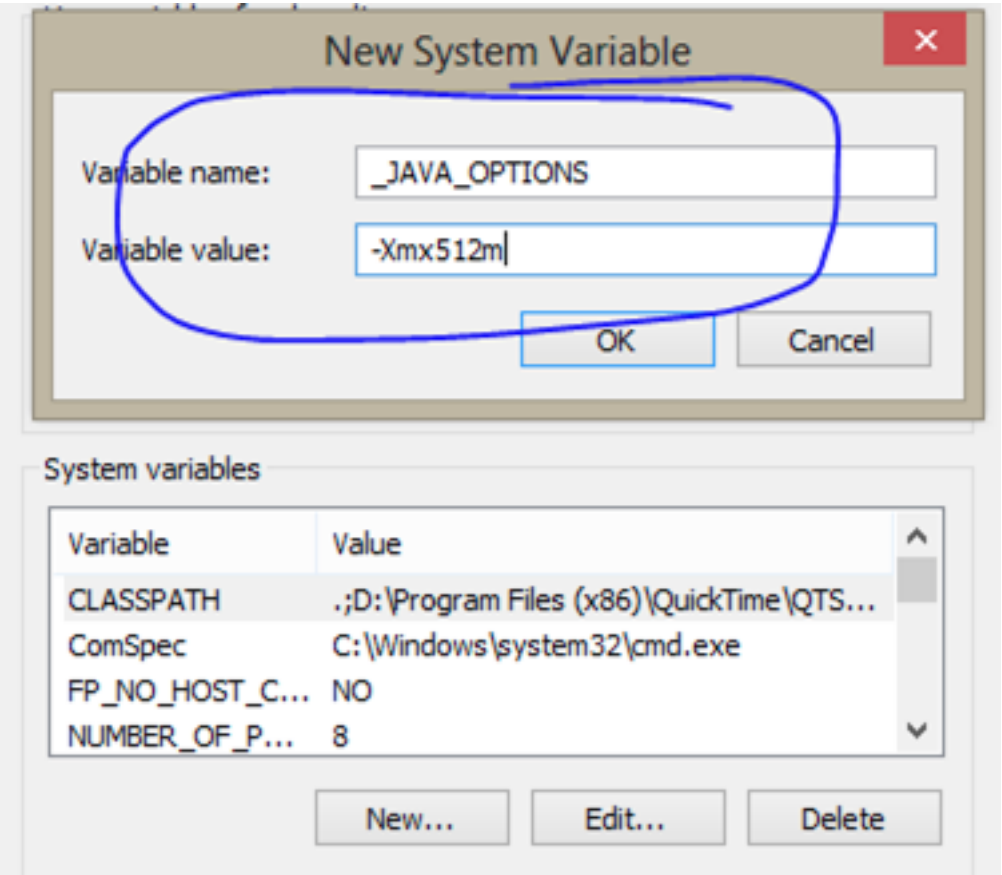
Stage	Who?	What?	Why?
Build	CI	WAR / JAR / etc	Avoid "It works in my machine"
Release	CD	Container image	Deployments / Updates and Rollbacks
Run	Platform	Container instance	Speed, Management, Orchestration

DEMO

Red Hat

3 - Config

Store config in the environment



Red Hat

3 - **Config** (What does it mean?)

If you have to repackage your application, you're doing it wrong!

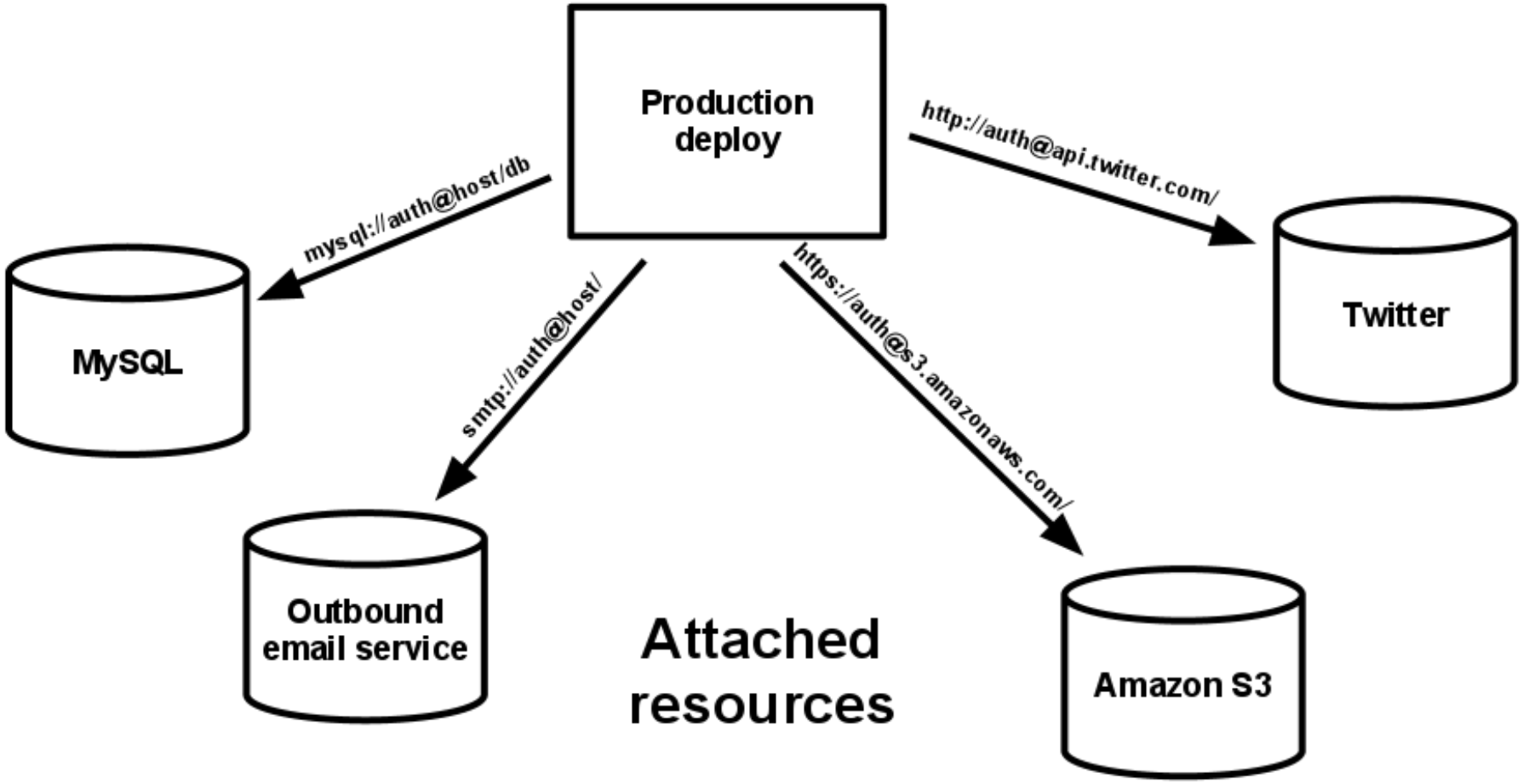
Prefer to store the config in Environment Variables

DEMO

Red Hat

4 - Backing Services

Treat backing services as attached resources



DEMO

Red Hat

6 - **Processes** (What does it mean?)

Execute the app as one or more stateless processes

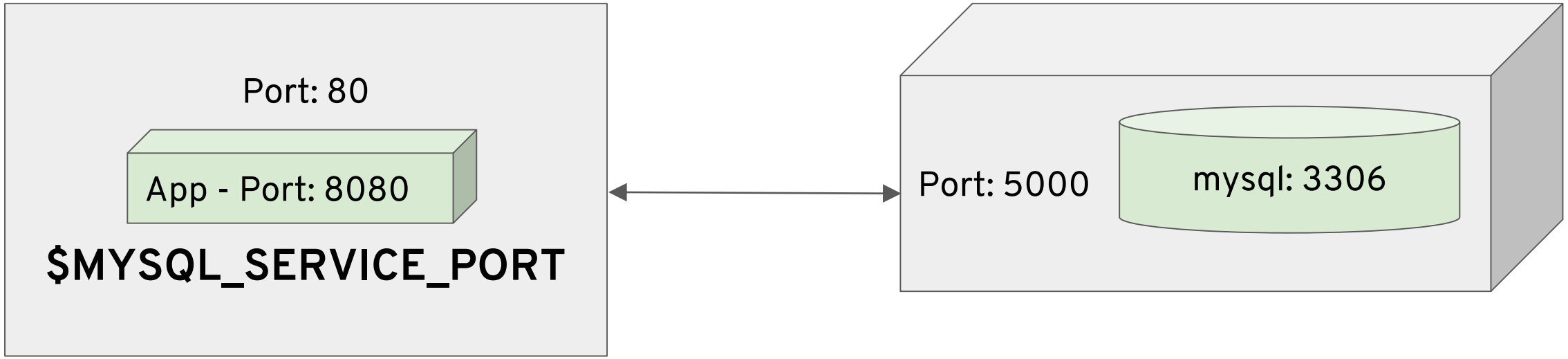
Twelve-factor processes are stateless and share-nothing

Red Hat

7 - **Port Binding** (What does it mean?)

Export services via port binding

The twelve-factor app is completely self-contained

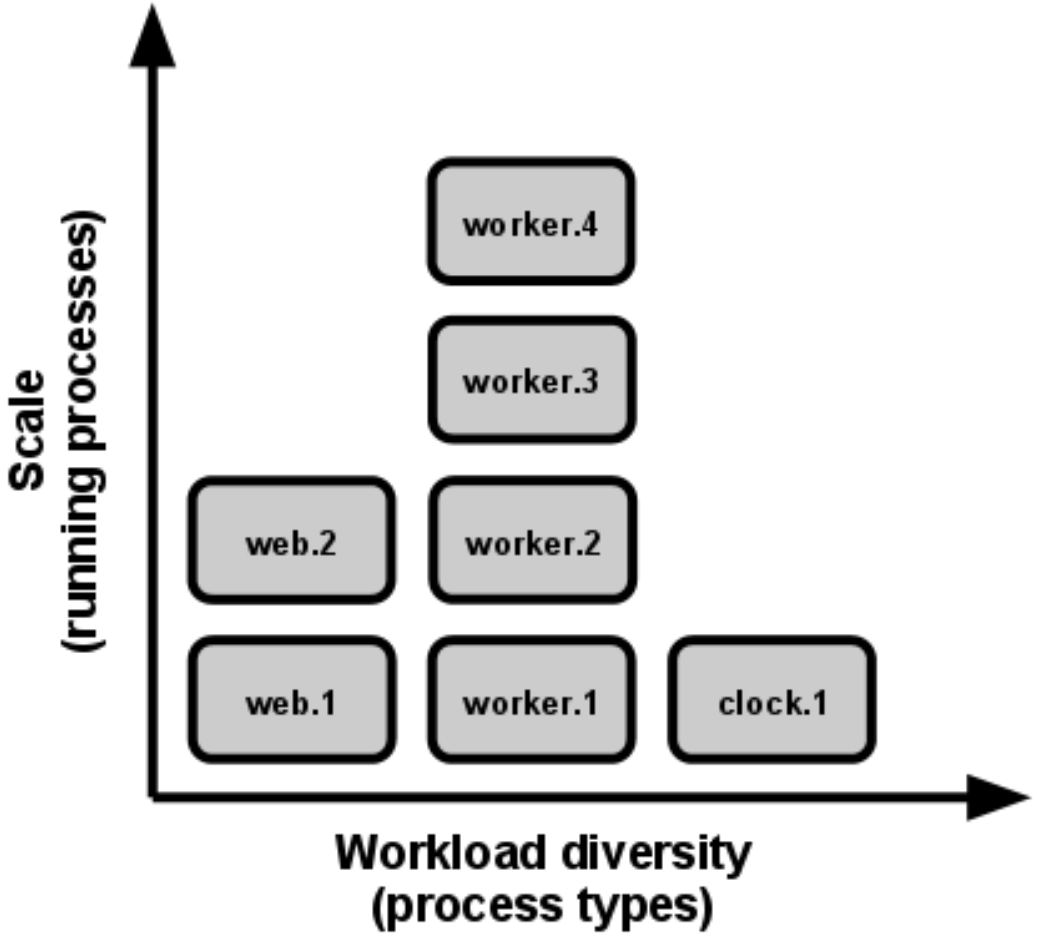


DEMO

Red Hat

8 - Concurrency

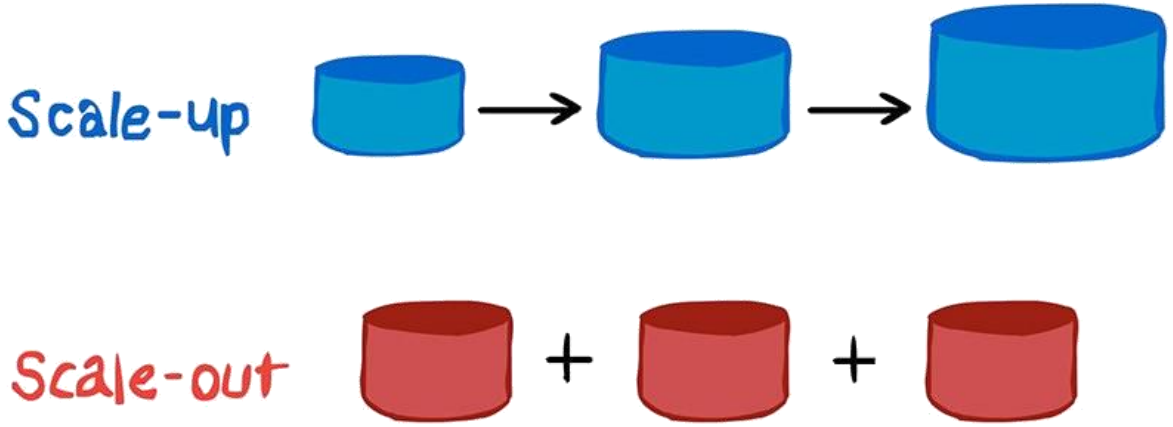
Scale out via the process model



Red Hat

8 - Concurrency

- You can scale **up** and **out**
- Scale **processes** types
- Workload **diversity**
- It "*advocates*" for **Microservices**



DEMO

Red Hat

9 - Disposability

Maximize robustness with fast startup and graceful shutdown

- Processes can be **started or stopped at a moment's notice**
- Processes should **minimize startup time**
- Processes **shutdown gracefully when they receive a SIGTERM**
- Processes **should also be robust against sudden death**

- You cannot **scale, deploy, release, recover fast** if you cannot **start fast**!
- You **cannot start** if you did not **shutdown gracefully**!

DEMO

Red Hat

10 - Dev/prod parity

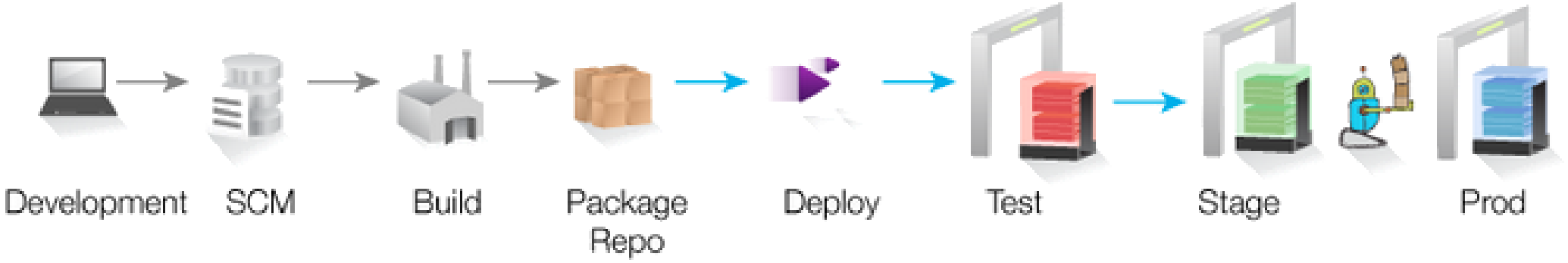
Keep development, staging, and production as similar as possible

Migrating manually directly to staging / production - **not** a great idea.

Red Hat

10 - Dev/prod parity

The twelve-factor app is designed for continuous deployment by keeping the gap between development and production small



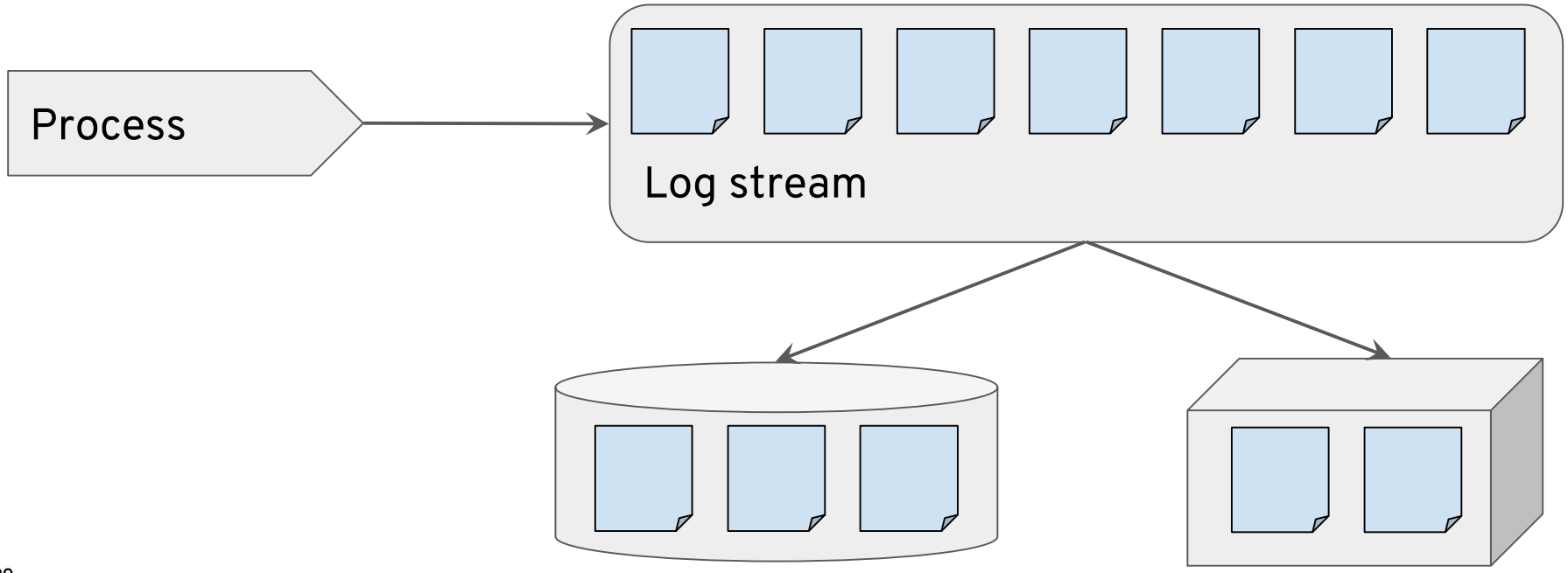
DEMO

Red Hat

11 - Logging

Treat logs as event streams

A twelve-factor app **never** concerns itself with **routing** or **storage** of its **output stream**

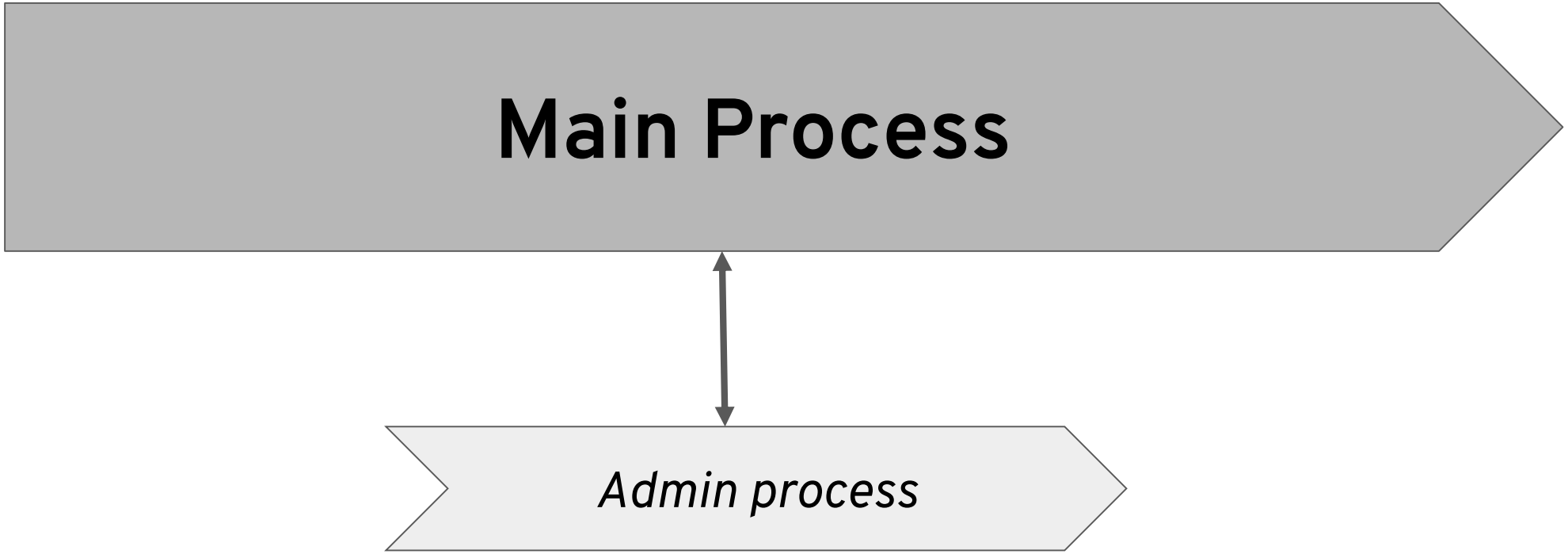


DEMO

Red Hat

12 - Admin processes

Run admin/management tasks as one-off processes



DEMO

Red Hat

Summary

- Methodology: Technology and language agnostic
- OpenShift: Technology and language agnostic
- But satisfied by
  - Containers,
  - Microservices,
  - and CI/CD Pipelines
- Focused on DevOps

- More info: https://12factor.net/
- Demo app: https://github.com/rbaumgar/12factor-app

Red Hat

Red Hat | Customer Portal | Developer | Hosted IDE | Marketplace | OpenShift | Partner Connect | Log In

Red Hat Developer

http://developers.redhat.com

Build | Tools | Events | Training | Partners | Products

Istio | Quarkus | Kubernetes | CI/CD | Serverless | Java | Linux | Microservices | DevOps | All topics

Latest articles

- Open Data Hub 0.6 brings component updates and Kubeflow architecture
- Using Ansible to automate Google Cloud Platform
- Working with big spatial data workflows (or, what would John Snow do?)
- Monitor business metrics with Red Hat Process Automation Manager, Elasticsearch, and Kibana
- Automated API testing for the KIE Server
- Application deployment improvements in OpenShift 4.4
- Serverless applications made faster and simpler with OpenShift Serverless GA
- OpenShift 4.4: Finding components in the Topology view

New OpenShift tutorials

Get started with OpenShift 4, log into a cluster, deploy applications from images or source, and experiment with your own code using the web console or command line.

View tutorials

Red Hat OpenShift 4

Red Hat Summit 2020 is an on-demand experience. Watch here,

Join developers across the globe for live and virtual events.

Open source exists for times like these.

Learn about developer tools

Thank you

linkedin.com/company/Red-Hat

facebook.com/RedHatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat