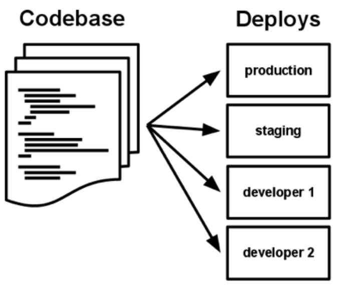
CLOUD NATIVE

12-Factor Applications

- Methodology for SaaS applications

Factor 1: Codebase

- Have one codebase tracked in revision control, many deploys
  - One repo for all code
  - Multiple deploys (instances, environments)
  - Includes *Infrastructure as Code* (IaC)



50

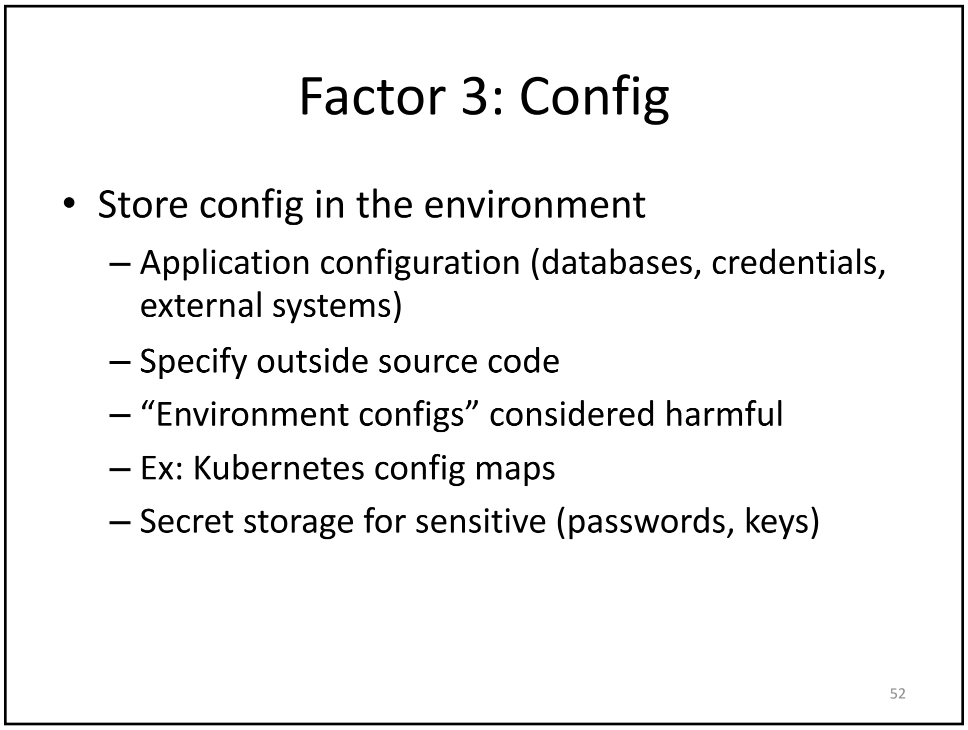
50

Factor 2: Dependencies

- Explicitly declare and isolate dependencies
  - 3rd party libraries (Maven)
  - Runtime, operating system (Docker)
    - Don’t use Docker “latest” tag
- Reproducibility
- Compatibility

51

51

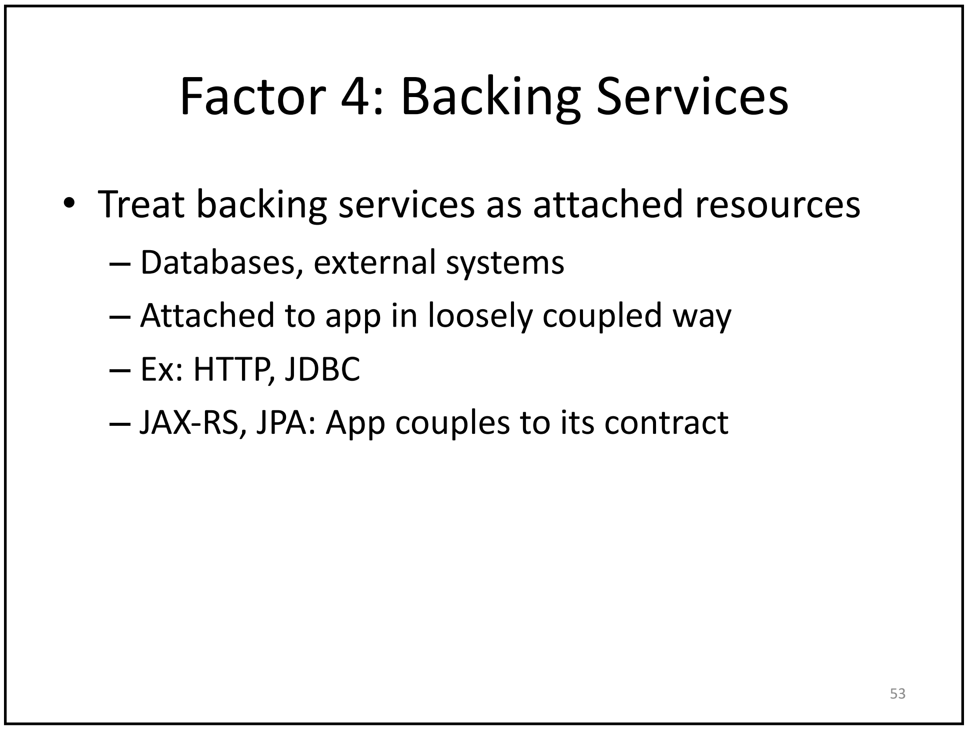


Factor 3: Config

- Store config in the environment
  - Application configuration (databases, credentials, external systems)
  - Specify outside source code
  - “Environment configs” considered harmful
  - Ex: Kubernetes config maps
  - Secret storage for sensitive (passwords, keys)

52

52



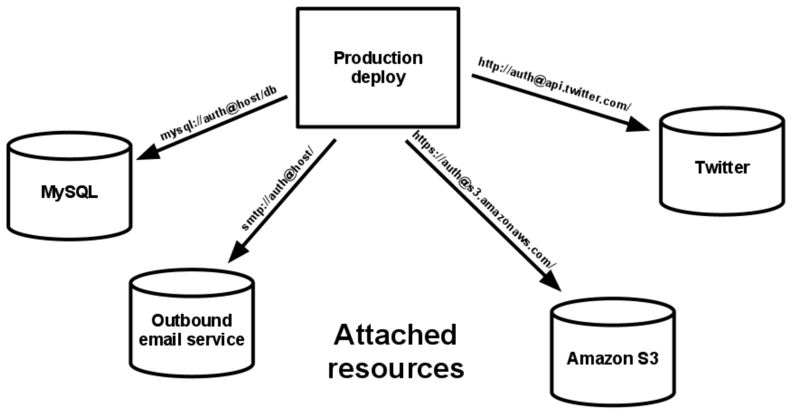
Factor 4: Backing Services

- Treat backing services as attached resources
  - Databases, external systems
  - Attached to app in loosely coupled way
  - Ex: HTTP, JDBC
  - JAX-RS, JPA: App couples to its contract

53

53

Factor 4: Backing Services

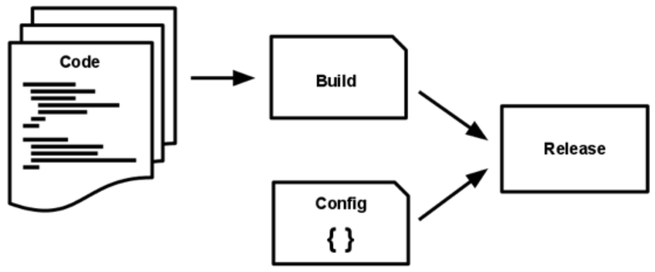


54

54

Factor 5: Build, Release, Run

- Strictly separate build, deploy and run stages
  - Deployment combines binaries and config
  - Orchestrate stages in Continuous Integration Server



55

55

Factor 6: Processes

- Execute the app as one or more stateless processes
  - Store state in attached resource (database)
  - Avoid session state in server
  - Docker copy-on-write file system
  - Stateless session beans

56

56

Factor 7: Port binding

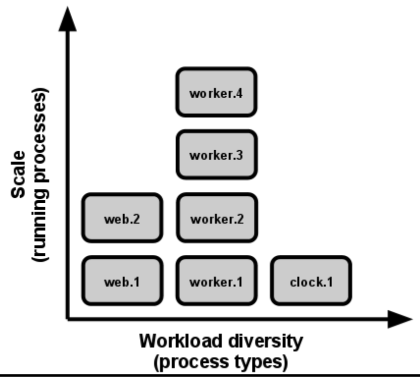
- Export services via port binding
  - Least coupling
  - Ex: Jetty Web server in Java app

57

57

Factor 8: Concurrency

- Scale out via the process model
  - Horizontal rather than vertical scaling
  - Self-contained *share-nothing* processes



58

58

Factor 9: Disposability

- Maximize robustness with fast startup and graceful shutdown
  - Fast startup for scalability
  - Graceful shutdown (Unix signals)

```
@ApplicationScoped
public class CoffeePurchaser {
   private Client client;
   ...
   @PreDestroy
   public void closeClient() {
      client.close();
   }
}
```

59

59

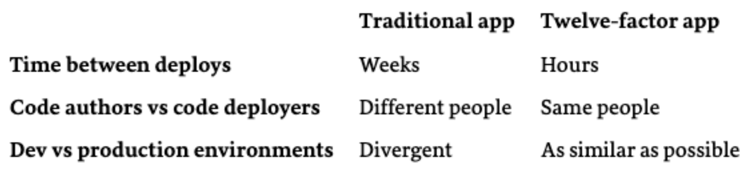
Factor 10: Dev/Prod Parity

- Keep development, staging, and production as similar as possible
  - Development workstations vs server clusters
  - Development team vs production team
  - Containers, orchestration frameworks for uniform environments
  - Continuous Delivery (min time to production) to remove differences between teams
  - DevOps

60

60

Factor 10: Dev/Prod Parity



	Traditional app	**Twelve-factor app**
Time between deploys	Weeks	Hours
Code authors vs code deployers	Different people	Same people
Dev vs production environments	Divergent	As similar as possible

61

61

Factor 11: Logs

- Treat Logs as Event Streams
  - Useful for app monitoring
  - Write to output (avoid logging dependencies)

62

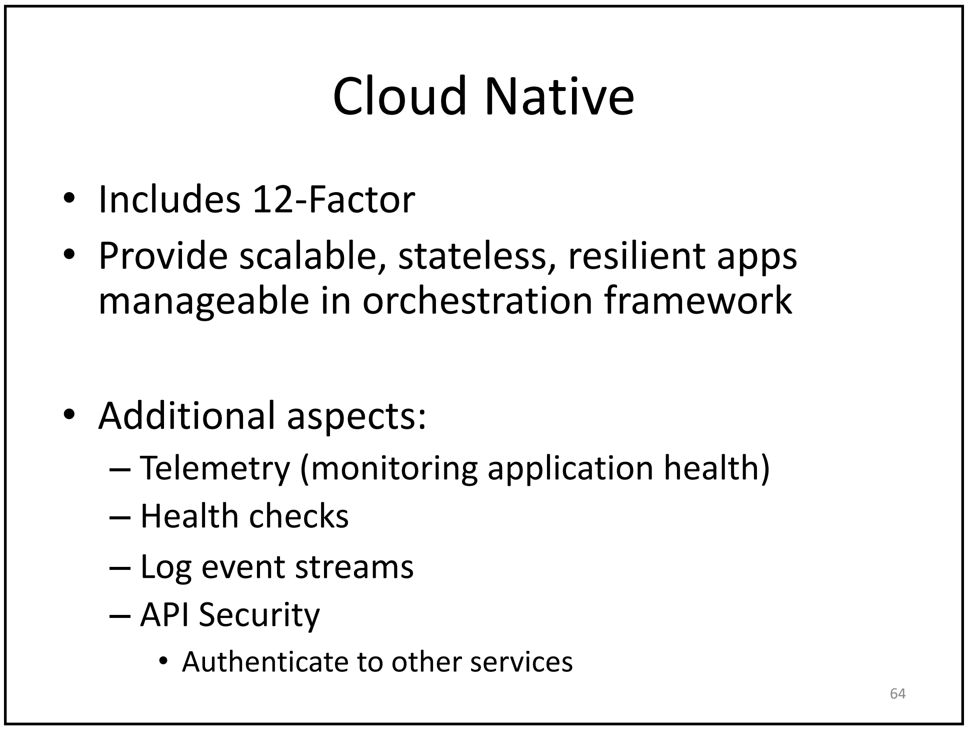
62

Factor 12: Admin Processes

- Run admin/management tasks as one-off processes
  - Debugging, trouble-shooting
  - Containers: can open remote shell into container

63

63



Cloud Native

- Includes 12-Factor
- Provide scalable, stateless, resilient apps manageable in orchestration framework

- Additional aspects:
  - Telemetry (monitoring application health)
  - Health checks
  - Log event streams
  - API Security
    - Authenticate to other services

64

64