IBM Cloud and 12-factor Apps

IBM CODE

What are the Twelve Factors?

https://12factor.net/

1. Codebase - One codebase tracked in revision control, many deploys
   - 1-1 relationship between app & code repo – use packages for shared code
2. Dependencies – Declared and isolated (no system wide dependencies)
3. Config - Store config in the environment (not in constants in the app)
4. Backing Services - Treat backing services as attached resources
   - Can be attached and reattached w/o affecting code, no differentiation between local and remote
5. Build, release, run - Strictly separate build and run stages
   - Release has unique id
6. Processes - Execute the app as one or more stateless processes
   - State shared via external services – no sticky sessions !
7. Port binding - Export services via port binding
   - In deployment, a routing layer handles routing requests from a public-facing hostname
8. Concurrency - Scale out via the process model
9. Disposability - Maximize robustness with fast startup and graceful shutdown
10. Dev/prod parity - Keep development, staging, and production as similar as possible
11. Logs - Treat logs as event streams
12. Admin processes - Run admin/management tasks as one-off processes

IBM Cloud Services / April, 2018 / © 2018 IBM Corporation

Twelve Factor and IBM Cloud - 1. Codebase

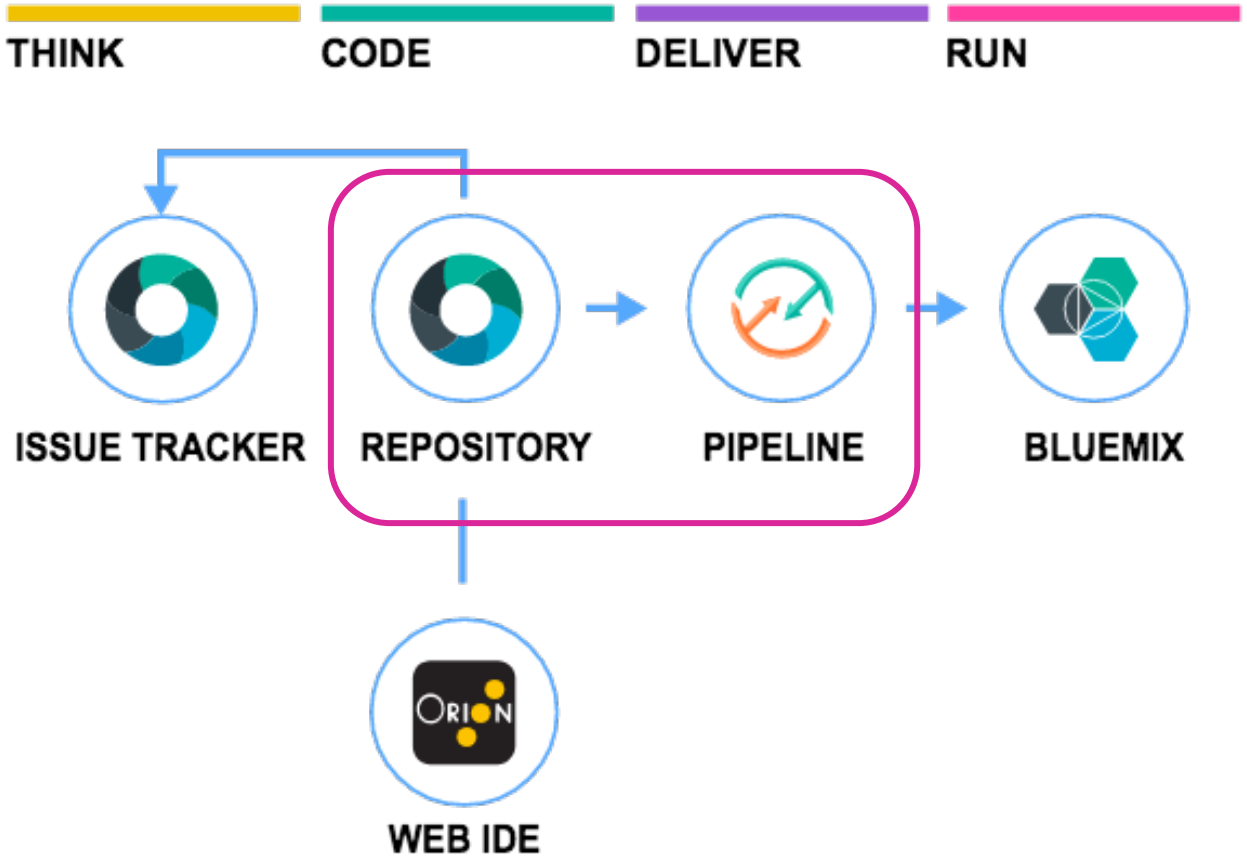
1. **Codebase**
2. Dependencies
3. Config
4. Backing Services
5. Build, release, run
6. Processes
7. Port binding
8. Concurrency
9. Disposability
10. Dev/prod parity
11. Logs
12. Admin processes

- One codebase tracked in revision control, many deploys

- **Cloud Foundry: utilize IBM Continuous Delivery toolchains or external automation with Cloud Foundry tooling (Urban Code Deploy, Gradle, Jenkins, ...)**

IBM CODE

Codebase – IBM Cloud
Continuous Delivery toolchains



A *toolchain* is a set of tool integrations that support development, deployment, and operations tasks.

Tool integrations with the source code repository and delivery pipelines can drive multiple deployments from a single repository

IBM CODE

Twelve Factor and IBM Cloud - 2. Dependencies

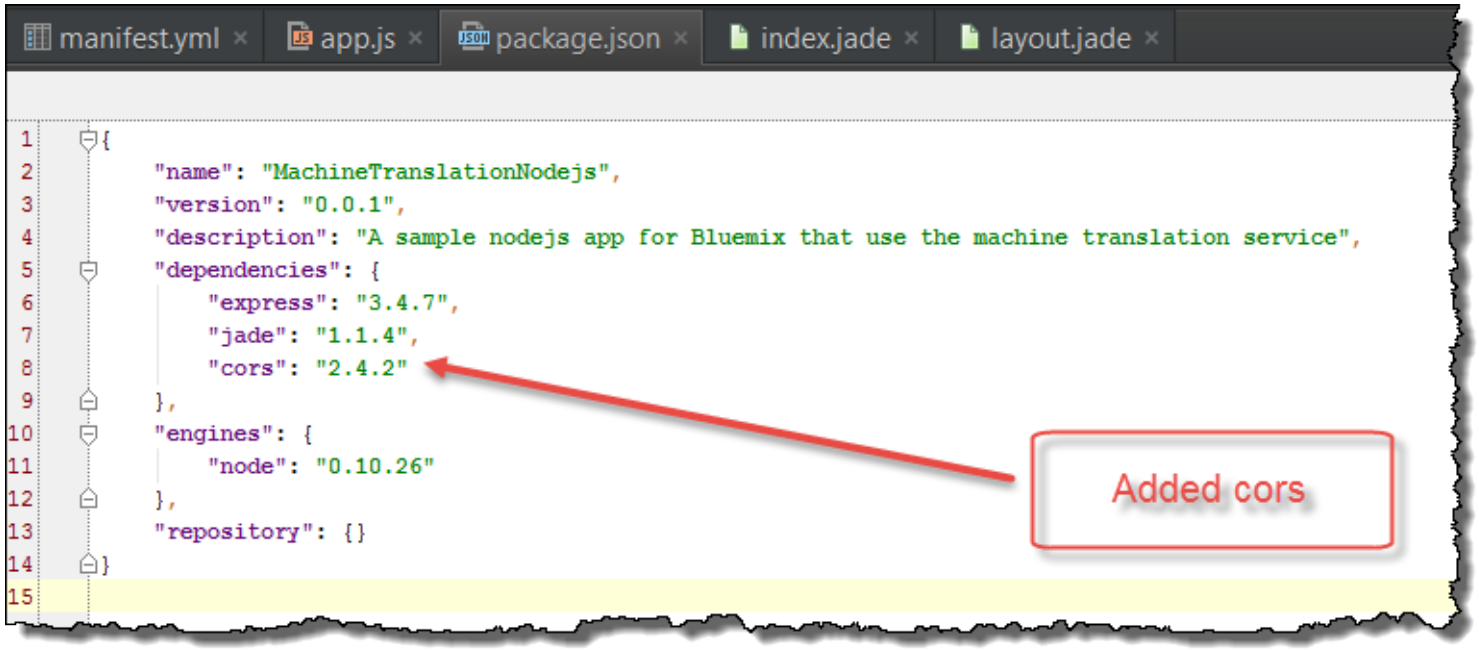
1. Codebase
2. **Dependencies**
3. Config
4. Backing Services
5. Build, release, run
6. Processes
7. Port binding
8. Concurrency
9. Disposability
10. Dev/prod parity
11. Logs
12. Admin processes

- Explicitly declare and isolate dependencies
- Typically platform dependent e.g. npm, bundler, or Liberty feature manager
- Never rely on or assume system-wide dependencies

- **Cloud Foundry: buildpacks manage external dependencies during staging**

IBM CODE

Dependencies



IBM CODE

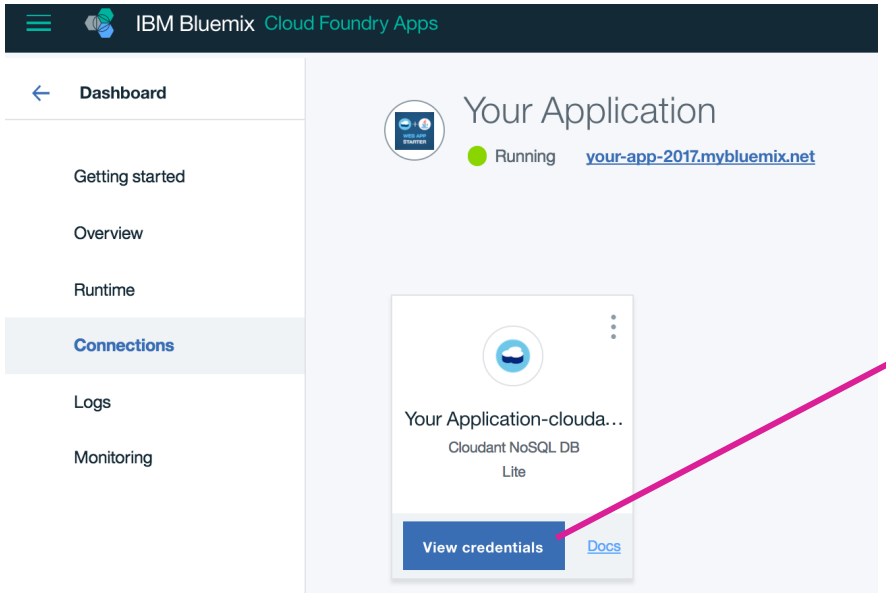
Twelve Factor and IBM Cloud - 3. Config

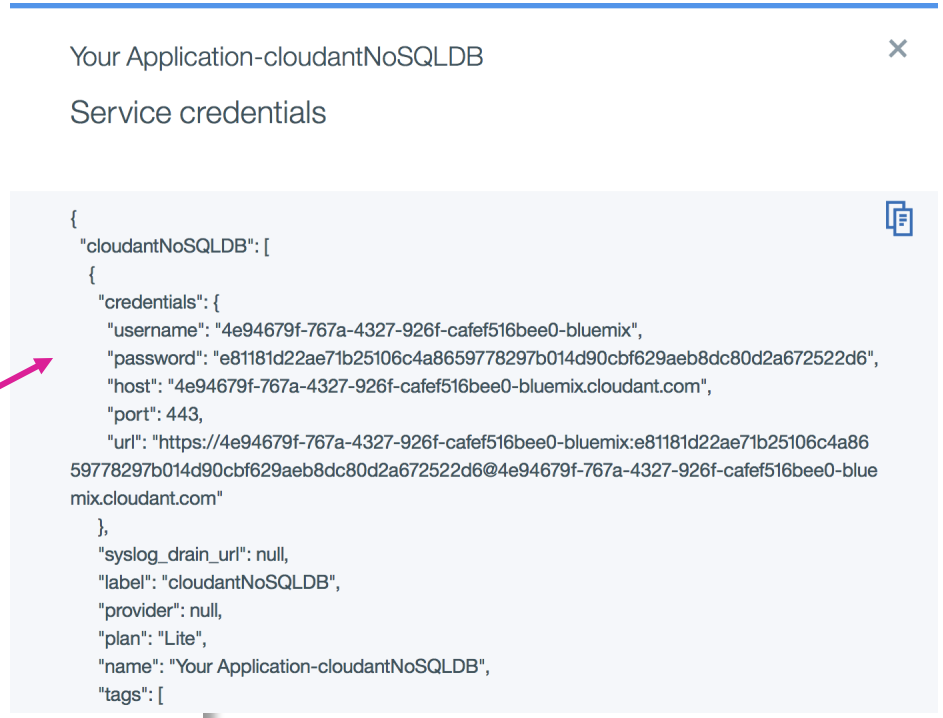
1. Codebase
2. Dependencies
3. **Config**
4. Backing Services
5. Build, release, run
6. Processes
7. Port binding
8. Concurrency
9. Disposability
10. Dev/prod parity
11. Logs
12. Admin processes

- Store config in the environment
- Separate config from source

- **Cloud Foundry: applications are parameterized via system provided and custom environment variables.**

Config






```
37    private static CloudantClient createClient() {
38        String VCAP_SERVICES = System.getenv("VCAP_SERVICES");
39        String serviceName = null;
40
41        if (VCAP_SERVICES != null) {
42            // When running in Bluemix, the VCAP_SERVICES env var will have the credentials for all b
43            // Parse the VCAP JSON structure looking for cloudant.
44            JsonObject obj = (JsonObject) new JsonParser().parse(VCAP_SERVICES);
45            Entry<String, JsonElement> dbEntry = null;
46            Set<Entry<String, JsonElement>> entries = obj.entrySet();
47            // Look for the VCAP key that holds the cloudant no sql db information
48            for (Entry<String, JsonElement> eachEntry : entries) {
49                if (eachEntry.getKey().toLowerCase().contains("cloudant")) {
50                    dbEntry = eachEntry;
51                    break;
52                }
53            }
54            if (dbEntry == null) {
55                throw new RuntimeException("Could not find cloudantNoSQLDB key in VCAP_SERVICES env v
56            }
```

IBM CODE

Twelve Factor and IBM Cloud - 4. Backing Services

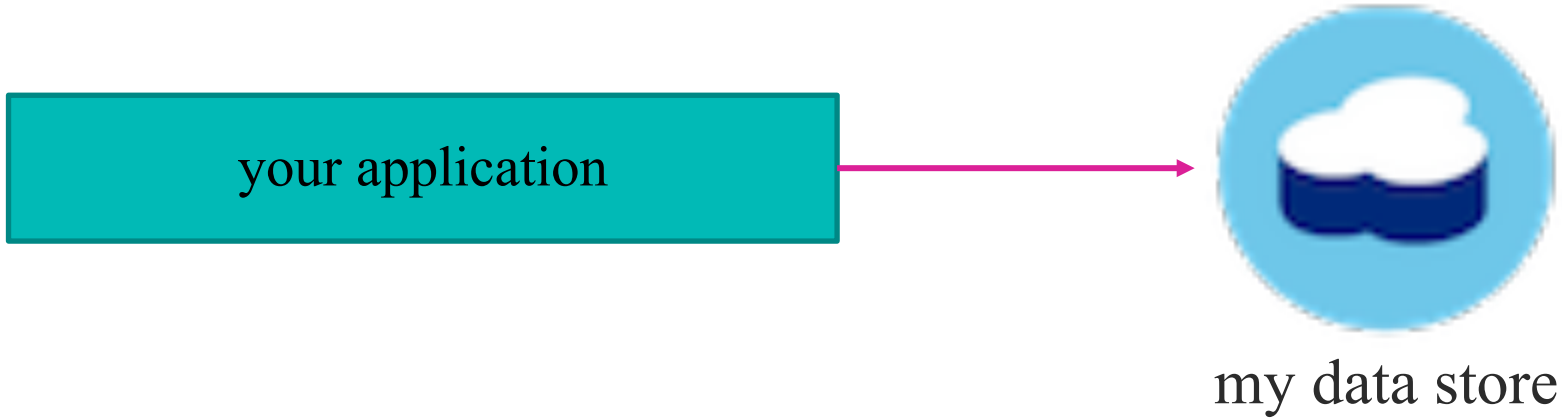
1. Codebase
2. Dependencies
3. Config
4. **Backing Services**
5. Build, release, run
6. Processes
7. Port binding
8. Concurrency
9. Disposability
10. Dev/prod parity
11. Logs
12. Admin processes

- Treat backing services as attached resources
- Local and remote resources should be treated identically

- **IBM Cloud: same mechanism for creating and binding to all services (including custom/external through user-provided services)**

IBM Cloud Services / April, 2018 / © 2018 IBM Corporation

Backing Services




```
ibmcloud app push "your application"--no-start
ibmcloud service create cloudantNoSQLDB Lite "my data store"
ibmcloud service bind "your application" "my data store"
ibmcloud app start "your application"
```

IBM CODE

Twelve Factor and IBM Cloud - 5. Build, release, run

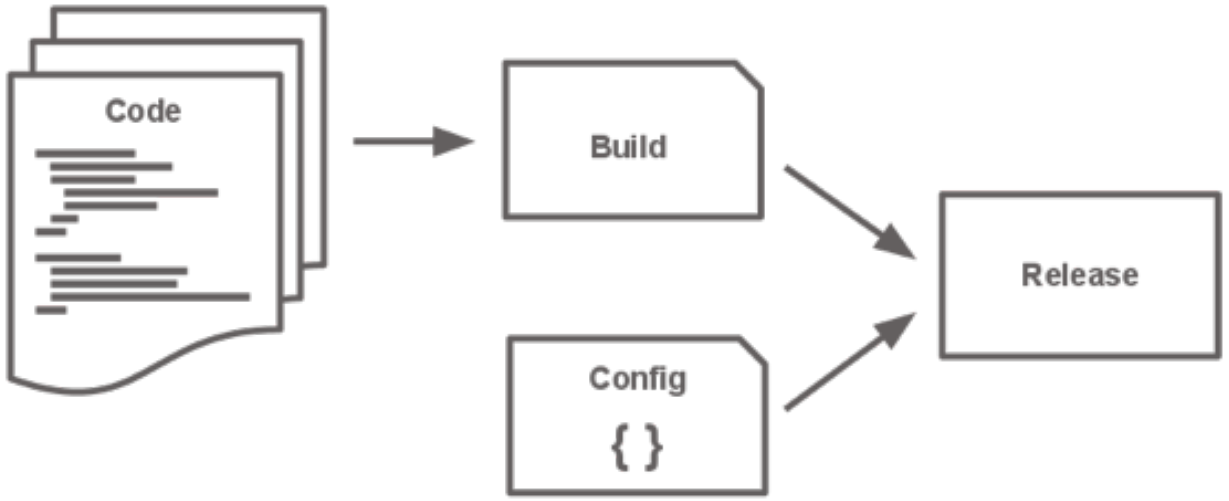
1. Codebase
2. Dependencies
3. Config
4. Backing Services
5. **Build, release, run**
6. Processes
7. Port binding
8. Concurrency
9. Disposability
10. Dev/prod parity
11. Logs
12. Admin processes

- Strictly separate build and run stages

- **Output from Cloud Foundry application build and staging is immutable container object. In IBM Cloud Container Service, the Docker build creates a container image that is stored to the private image registry.**

IBM CODE

Build, Release, Run



Code + CF Buildpack => immutable Garden container image

Code + Docker build => container image in private registry

Twelve Factor and IBM Cloud - 6. Processes

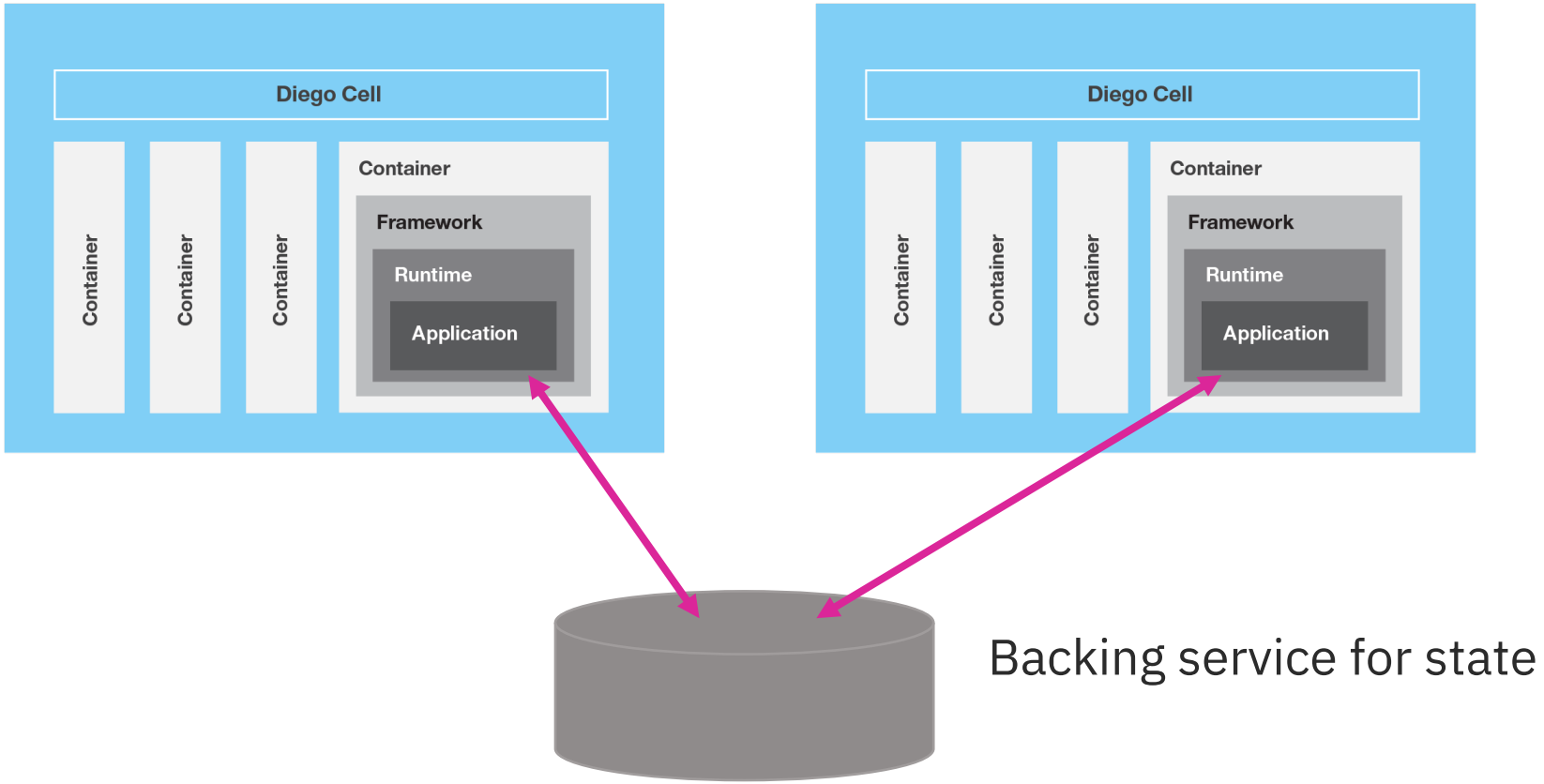
1. Codebase
2. Dependencies
3. Config
4. Backing Services
5. Build, release, run
6. **Processes**
7. Port binding
8. Concurrency
9. Disposability
10. Dev/prod parity
11. Logs
12. Admin processes

- Execute the app as one or more stateless processes
- Never rely on sticky sessions
- **IBM Cloud: design application instances to be stateless (state is held by services)**

IBM CODE

Processes

Stateless : use external service to retain state



Twelve Factor and IBM Cloud - 7. Port Binding

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

- Export services via port binding

- **Cloud Foundry applications create a service port implementing HTTP or web sockets protocol. The IBM Cloud infrastructure handles routing of requests to the port. In IBM Cloud Container Service, services are declared and mapped as needed to the application in the container.**

IBM CODE

Twelve Factor and IBM Cloud - 8. Concurrency

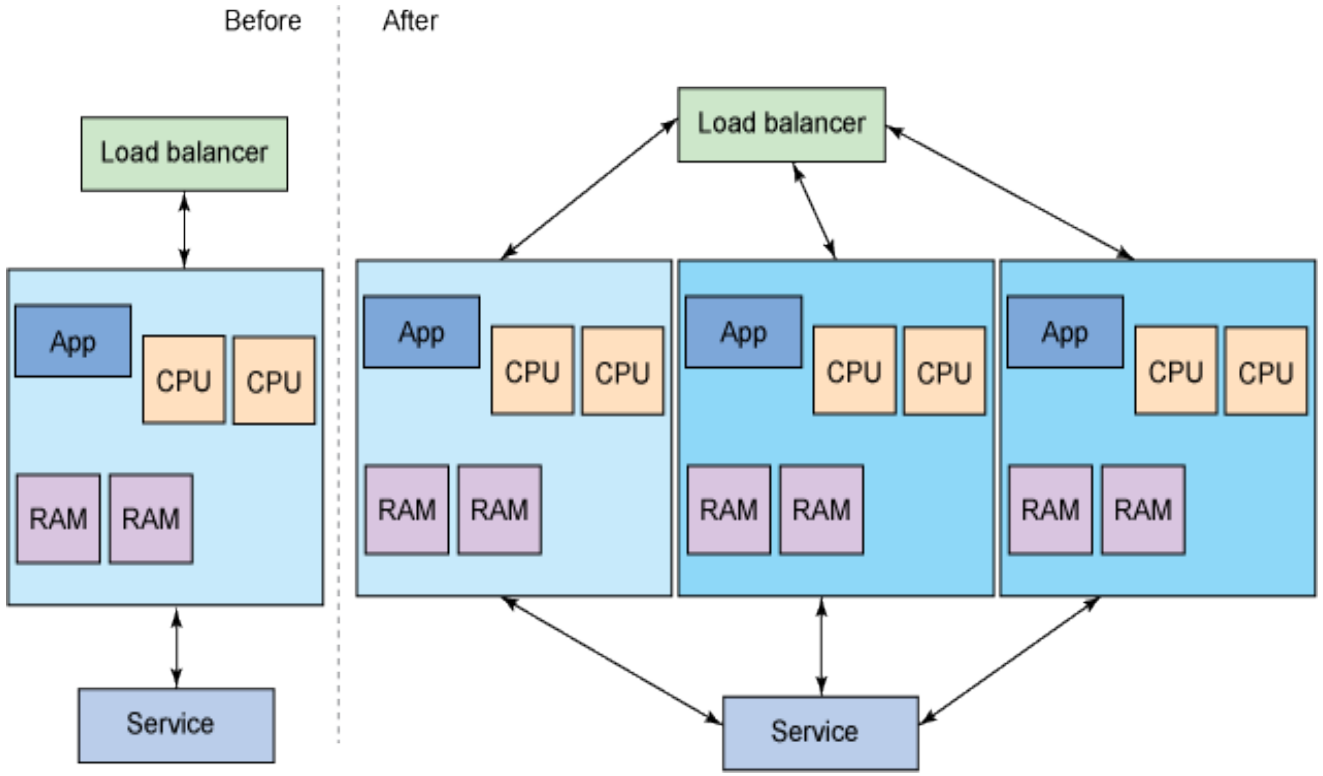
1. Codebase
2. Dependencies
3. Config
4. Backing Services
5. Build, release, run
6. Processes
7. Port binding
8. **Concurrency**
9. Disposability
10. Dev/prod parity
11. Logs
12. Admin processes

- Scale out via the process model
- Servers, VMs can only scale vertically so far
- Stateless service model makes scaling simple

- **For Cloud Foundry applications, use CLI or web UI to manually scale and auto-scaling service to scale based on app metrics. Containers use scalable container groups or kubernetes autoscaling.**

IBM CODE

Concurrency



Twelve Factor and IBM Cloud – 9. Disposability

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

- Maximize robustness with fast startup and graceful (and quick) shutdown
- Application instances are disposable
- Robust against death of underlying resources

- **Both Cloud Foundry runtimes and IBM Cloud Container Service containers quickly start and terminate, but the application must adhere as well**

IBM CODE

Twelve Factor and IBM Cloud – 10. Dev/prod parity

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

- Keep development, staging, and production as identical as possible
- Use the same backing service types and versions in every environment

- **Cloud Foundry spaces can be used to separate environments all running in the same organization and hosting platform**

IBM CODE

Twelve Factor and IBM Cloud – 11. Logs

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

- Treat logs as event streams
- Don't write to log files

- **The Cloud Foundry loggregator provides event streams for applications; can be drained to third-party log management system.**

IBM CODE

Twelve Factor and IBM Cloud – 12. Admin processes

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**

- Run admin/management tasks as one-off processes
- E.g. database migrations or for debugging

- **Use separate single-shot admin processes bound to the same services as application**

IBM CODE

IBM
CODE