Cloud-specific software architecture patterns

What Motivation How When Watch out!

Packaged configuration


Configuration is packaged with deployment artefacts


Simplify system, increase resilience by removing runtime dependency on configuration service


Configuration is managed in configuration repository, CI/CD combines generic application artefact with stage/tenant-specific configuration and deploys it


Multiple stages / tenants, build pipeline is flexible


No runtime update of configuration, configuration changes require redeployment

Natural multi-tenancy


Each logical tenant and/or stage is an isolated installation


Simplify system by designing only for single tenant use


Each tenant runs in a dedicated and isolated environment


Groups of users requiring isolated setups


Problematic when users have access to multiple tenants

Swarm uptime


Combining uptime of multiple, unreliable service instances into high application uptime


Lower cost by using cheap, volatile instances


The cloud runtime sends tasks to running service instances, but not to failing instances. Failing instances are restarted automatically.


App instances don't rely on internal state, or that state can be restored easily. App instances boot quickly.


Containerised applications must be able to handle arbitrary restarts

https://blog.georgovassilis.com/2018/08/29/cloud-specific-software-architecture-patterns/

Cloud-specific software architecture patterns

What Motivation How When Watch out!

Automated maintenance

- Offload maintenance tasks to runtime platform
- Simplify application & project design
- The cloud runtime handles maintenance tasks transparently
- Periodic instance restarts to combat latent resource leaks, clean filesystem, backup, store logs, produce metrics
- Can't handle overly specific tasks

Outscale caching

- Achieve high application performance
- Simplify application design by avoiding the complexity of caching
- Scale service instances instead of caching data
- Caching (and cache invalidation) are overly complex for the domain, service instances don't rely on internal state, service invocation cascades are shallow, latency requirements can be met
- Potentially resource-hungry

Service discovery

- Broker between service instances
- Simplify application design by avoiding the complexity of service discovery
- The cloud platform injects service location information as part of "Packaged configuration" and routes requests dynamically at runtime
- Always
- None

https://blog.georgovassilis.com/2018/08/29/cloud-specific-software-architecture-patterns/

Cloud-specific software architecture patterns

What Motivation How When Watch out!

Security and access control


Secure application easily


Simplify application design by delegating (more) access control to the cloud runtime


Less infrastructure is exposed, access is controlled by the platform


Always


Reduced flexibility

Infrastructure as code


Configure infrastructure through (versioned & audited) code


Simplify change management


Infrastructure as code allows defining entire systems through declarative scripts


In a cloud environment


Hard to get used to, slows down development, won't work on non-cloud-native infrastructure

https://blog.georgovassilis.com/2018/08/29/cloud-specific-software-architecture-patterns/