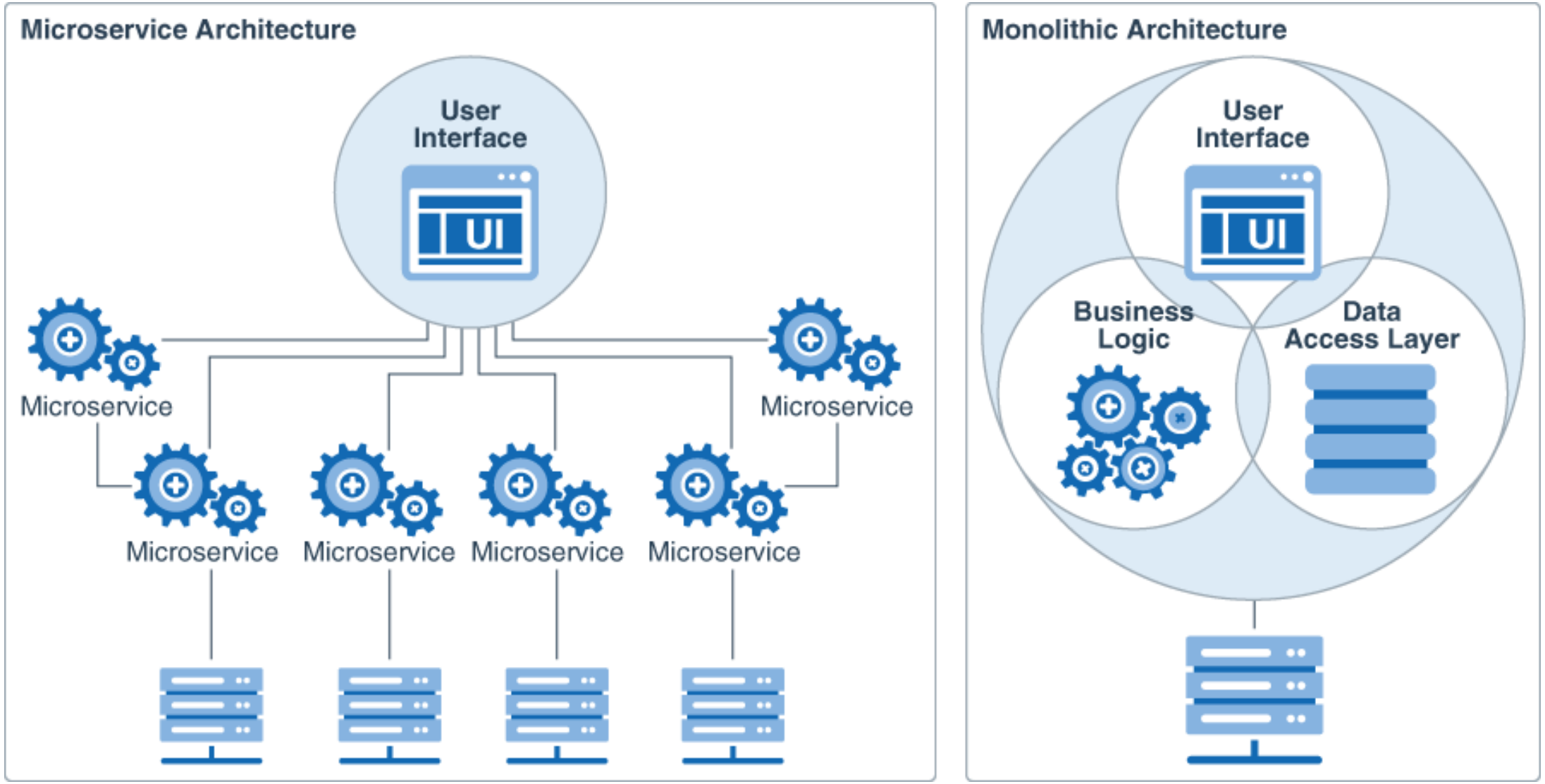
Microservices and Cloud Services

Greg Pavlik

SVP, Oracle Cloud Infrastructure

From Monolith to Services



What are services?

- Network accessible compute endpoints
- Commodity protocols
  - Network, data, security (reliability)
- Typically scalable and fault tolerant
- Long evolution from monolithic applications to ”service oriented” approaches
  - Many lessons learned….

Evolution of Services

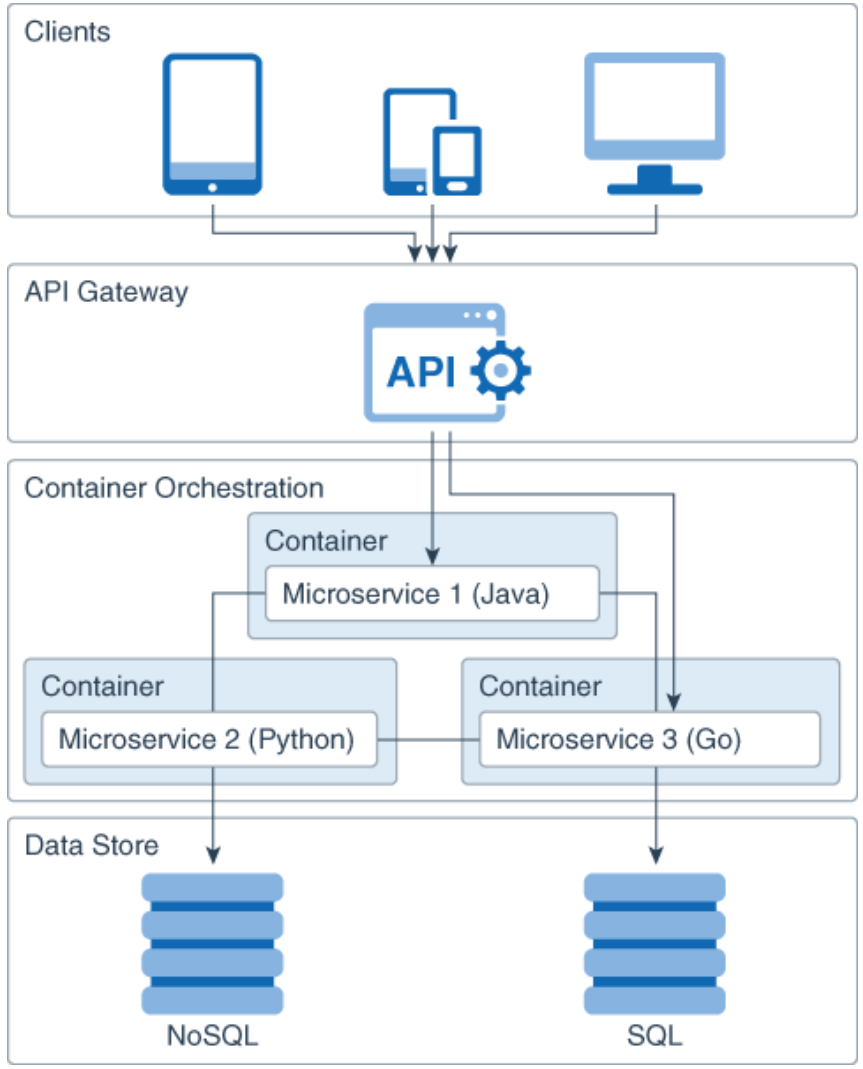
- Network endpoints
- Distributed Objects (CORBA, RMI)
- Application Servers
- Web Services
- Microservices

Universal Challenges

- Encapsulation
  - API design critical
- Factoring/Granularity
  - Goes beyond coherence and cohesive
  - Science meets art – design is still a craft
- Robustness
  - State management the key problem
  - Where many ad hoc approaches fail (significant flaw in early distributed systems)

Microservices – Characteristics

- Loosely coupled services, each supporting single business task.
- Microservices expose APIs which enables functionality reuse.
- Each microservice can choose its own programming language and framework.
- Decentralized data management. Each microservice can have its own database.
- Independent deployment, easier to maintain
- Distributed functionality leads to higher resilience
- Each microservice can be scaled independently



MicroServices – What's the big deal?

- Flexibility - Removes the implementation constraints of frameworks
  - Implement and consume from any language
  - Standardize at protocol layer
    - Usually basic HTTP and JSON
- Container based deployments work well for "real" CI/CD
  - Easy deployment/upgrade/downgrade (they should be the same thing)
- Largely open source based
  - Free vs 10,000s USD per node
  - Lots of investment
- Starting to tackle the hard problems
  - Orchestration and container lifecycle
  - Istio and other "service mesh" frameworks (distributed filters)

Microservices – Best practices?

- Applications vs Enterprise vs Cloud Scale
  - Back to the “Universal Challenges”
- Must think of the container as the basic unit of the microservice
- Responsive microservices: The microservices must return a response to the requesting clients, even when the service fails.
- Backward compatibility: The REST API must remain backward-compatible.
- Flexible communication: Same communication protocol must be used between the clients and the API gateway and between the microservices.
- Idempotency: If a client calls a microservice multiple times, then it should produce the same outcome.
- Service Mesh
  - Maybe. Or is this a sign of bad factoring?

Microservices – Challenges

- Complexity explosion
- Failure modes
- State management
- Network congestion and latency
- Data consistency
- Diagnostics

Cloud Services – What Are They?

- Internet accessible, functionally cohesive solutions
- Always API driven, sometimes use UI
- Highly available
  - Availability domains
- Multi-tenant security
- “Infinitely” scalable
  - Load and “instances”
- Composable to deliver higher levels of abstraction / functionality
  - Not a microservice pattern per se

Cloud Services – Architecture

- Separately scalable components of cloud service architecture
  - Control Plane – underlying service resource lifecycle management
  - Data Plane – infrastructure for customer workloads containing the service resources
  - Management Plane – service operations infrastructure for managing service operational health (bastion hosts, security scans, ops tools, etc.)
  - CICD bridges from development/test to production
- Internal use of
  - Microservices within a service control plane, data plane and management plane to implement the cloud service
  - Other lower level or adjacent cloud services to deliver integrated convenience or higher levels of functionality

Cloud Services – Best Practices

- Multi-tenancy considerations
- Security, compliance and governance
- Scale for data planes
  - Cell architecture used for “infinite” horizontal scale
- Cost and pricing
- Asynchronous processing
  - Cloud service responsible for managing resources safely and throttling as needed
- HA/DR
  - Failure modes – retry, degradation, outage
  - Blast radius – rack, data center, region; single tenant, all tenants

Cloud Services – Operations

- Majority of time spent on ops related work
  - Why are hundreds of people working on s3?
  - DevOps is standard model
- Want up to 5 9s
- Must see everything that happens
  - Dashboards
  - Metrics
  - Logs
  - Alarms
- Runbooks
- Self healing and self tuning are important!

Cloud Services – Intersections with Microservices

- Almost all cloud services use microservice patterns in Control Plane and Data Plane
- Never exposed outside service boundaries
  - Granularity and functionality are different
  - Load is controlled by service
  - Not always implemented with containers (“service principles”, security issues)
- Service Meshes almost never used
  - Less dense interaction patterns
  - The problem exists for cloud service to cloud service interactions, but side car pattern doesn’t work.

Conclusions

- Running on the cloud is not the same as delivering a cloud service
  - Way more than compute on the cloud…
- Well established patterns for how to do cloud services right
  - Most engineers don’t know them!
- Microservices used carefully can power cloud services
  - But don’t confuse the two
  - Whole companies have mis-steered engineering between monoliths and microservices