## Mrunal Kapure

### M.S. Software Engineering

### Arizona State University

### mkapure@asu.edu.

1. What do the authors mean by “infrastructure is fluid”? How can it be fluid?

When the authors say, “infrastructure is fluid,” they mean that the computers, networks, and storage that run cloud applications are always changing. In the cloud, things are flexible servers can be created or shut down in seconds, and apps might move from one machine to another without anyone noticing. This is different from traditional systems, where everything is fixed and predictable.

So, infrastructure can be fluid in the following ways:

1. It scales up or down depending on how much traffic there is.
2. It’s temporary, i.e., machines or containers can appear or disappear anytime.
3. It’s shared and spread across different regions and users.
4. You don’t manage physical machines directly. Instead, you use tools or services to handle it all.

2. What do they mean by “failure is constant”? How is failure constant?

“Failure is constant” means that in cloud systems, things go wrong all the time. Even though cloud providers build very reliable systems, the fact that you’re working at such a large scale means something is always broken somewhere.

Here are some ways failure might shows up:

1. A server or disk might crash.
2. A network connection might drop or slow down.
3. A software bug could cause something to crash.
4. A wrong configuration might break a part of the system.

Cloud-native apps are built with this in mind. They expect things to fail and have backup plans like retrying requests, using multiple copies of data across different availability zones, switching to another server automatically, or scaling up and down the infrastructure as per the requirements.

3. In Figure 1, what parts are true microservices, and what are appliances?

In the diagram, the article talks about:

1. The microservices are:
   1. M1 to M6: These scan the web for new articles as they’re independent and do one specific job.
   2. A1 to A5: These analyze the articles as each can be managed and scaled on its own.
2. The appliances are:
   1. AWS Simple Queueing Service (SQS): A built-in cloud service that holds messages between services.
   2. D1 to D3 (DynamoDB) and D’1 (CosmosDB): These are cloud databases managed by providers.

So, the microservices are the parts we build and control, while the appliances are trusted services provided by the cloud platforms.

4. What would be hard about building observability for this app (logs, metrics, traces)?

When it comes to making sure everything in this app is running smoothly, here’s what would be tough:

1. Logs:
   1. It's tricky to gather logs from many microservices, especially when they’re running in different places.
   2. Also, cloud services like SQS or DynamoDB might not give full access to their logs.
2. Metrics:
   1. Each service might report data differently, and combining all that info can be messy.
   2. It’s hard to get a full picture when parts of the system (like databases) are out of your control.
3. Traces:
   1. Tracing means following one request through the whole system. That’s really hard here because:
      1. The system uses both your own code and cloud services.
      2. Not every service is traceable — you can’t see what happens inside AWS SQS or CosmosDB.

Basically, observability is challenging because here they have a mix of their own services and external ones, and tying them all together into one clear view would be difficult.