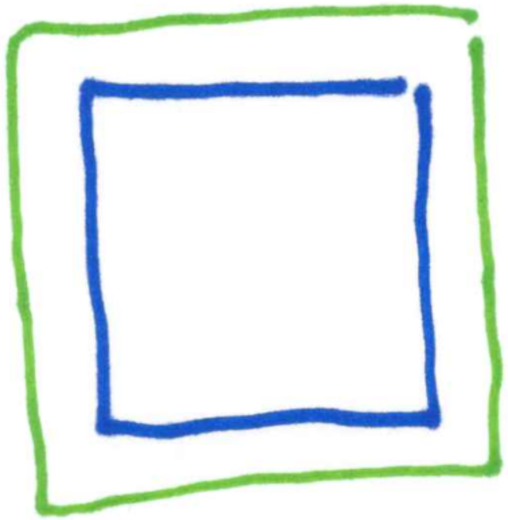
Microservices

Holly Cummins
@holly_cummins

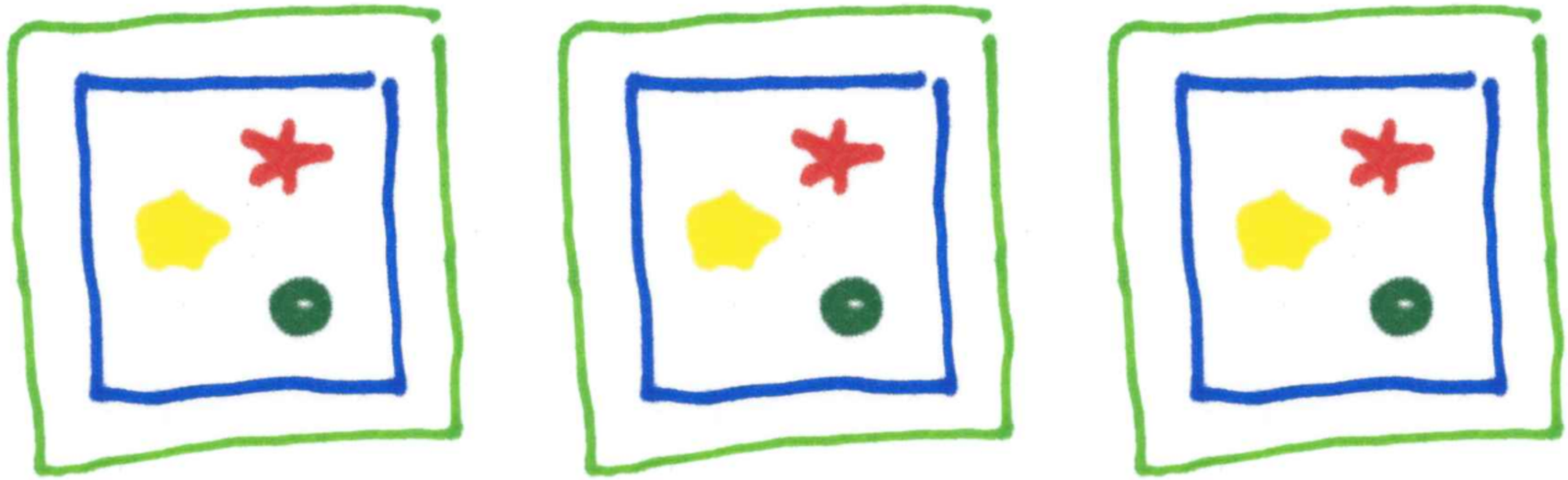

IBM
Dr. Holly Cummins
Software Engineer
Java Technology Centre
IBM Software Group
IBM United Kingdom Limited
Hursley Park
Winchester, Hampshire
SO21 2JN
cumminsh@uk.ibm.com
IBM Bluemix™
The Apache Software Foundation
http://www.apache.org/

http://ibm.biz/bluemixgaragelondon



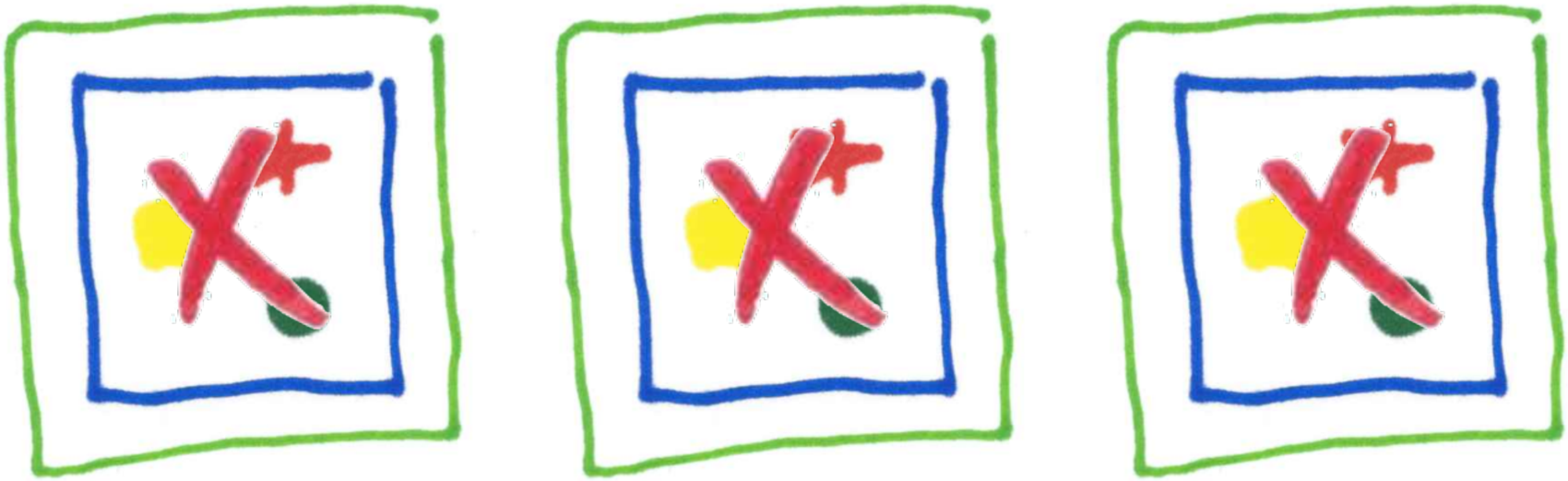
Modularity

Monolithic application



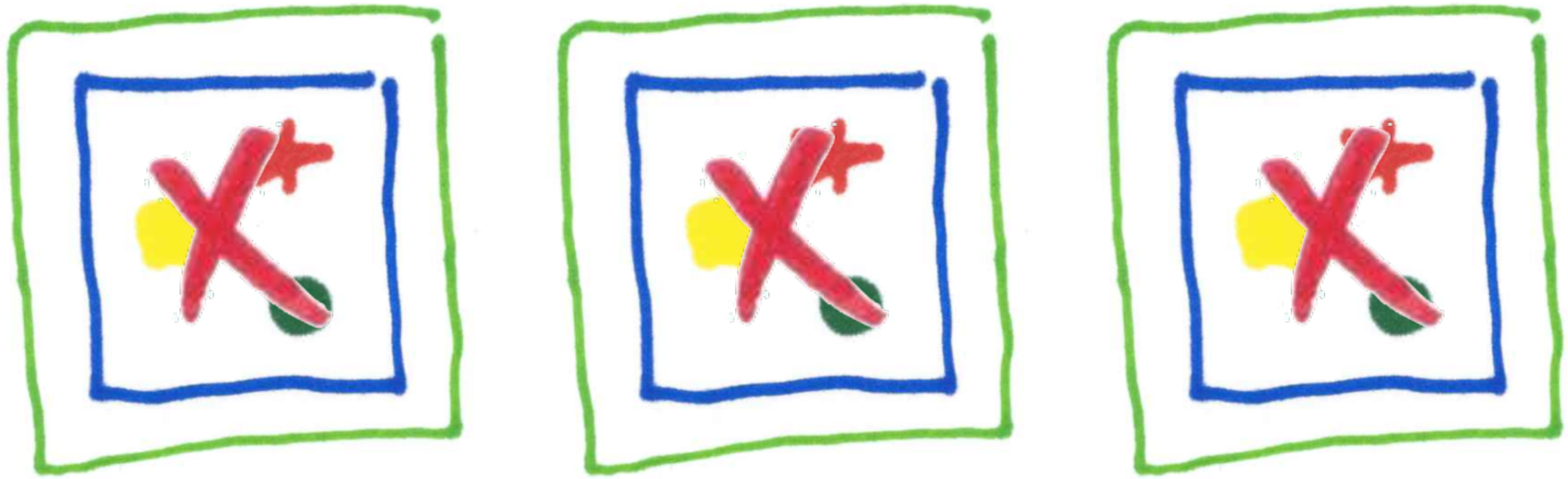
Scaling

Monolithic application



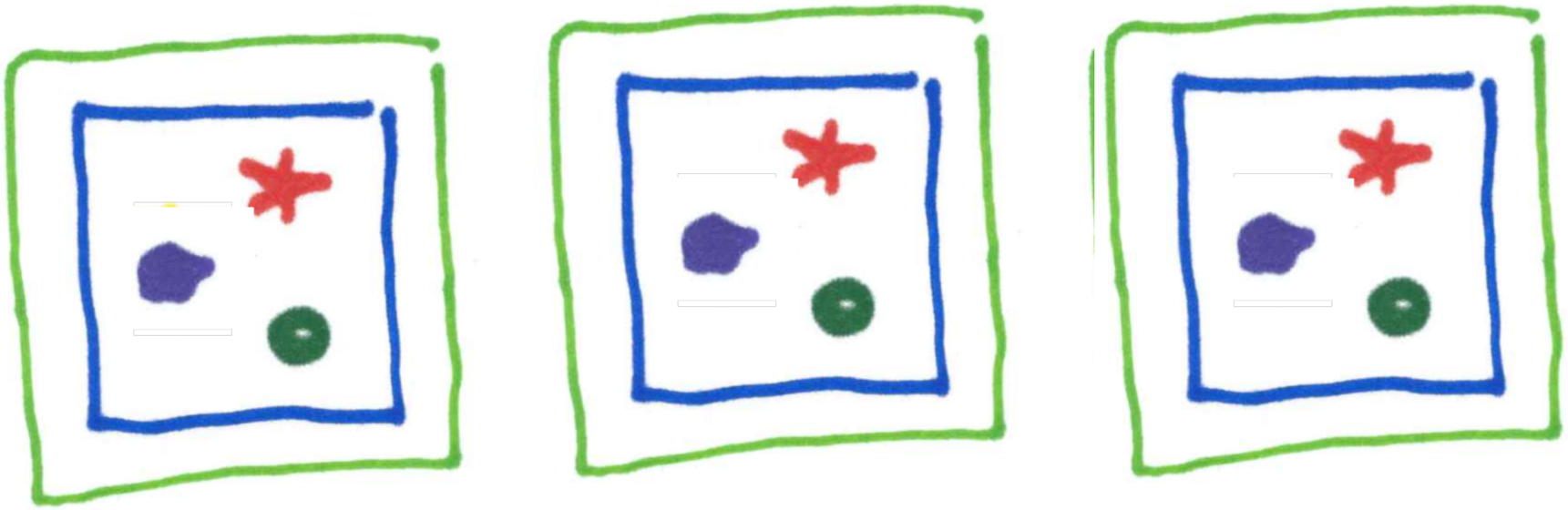
Failing

Monolithic application



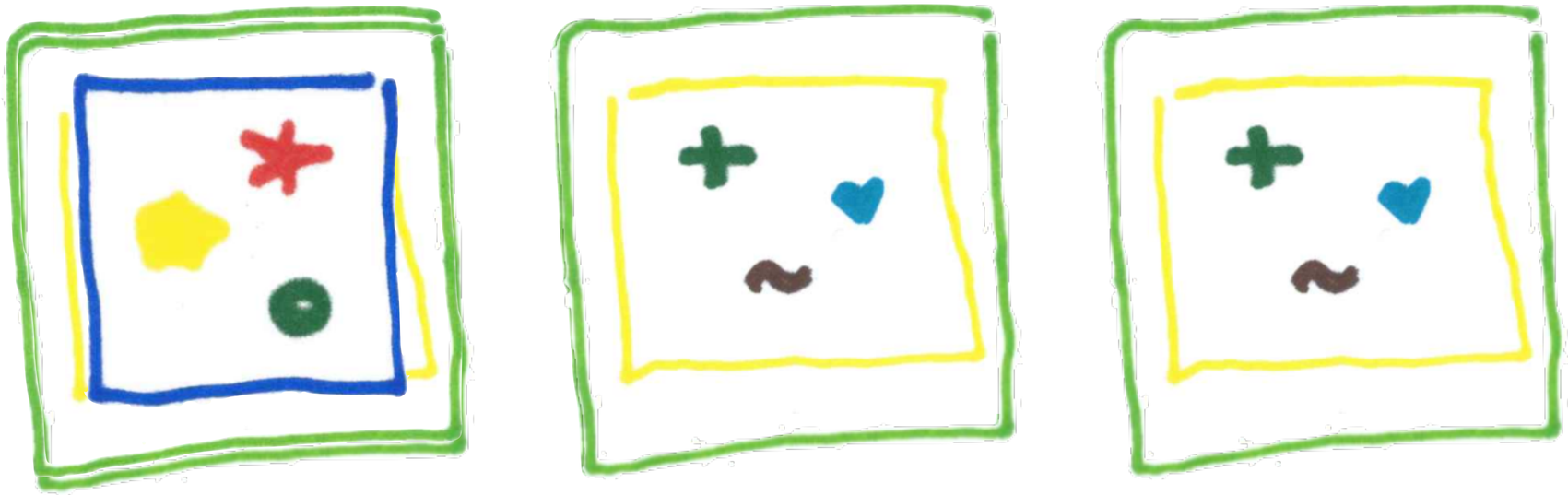
Failed

Monolithic application



Update

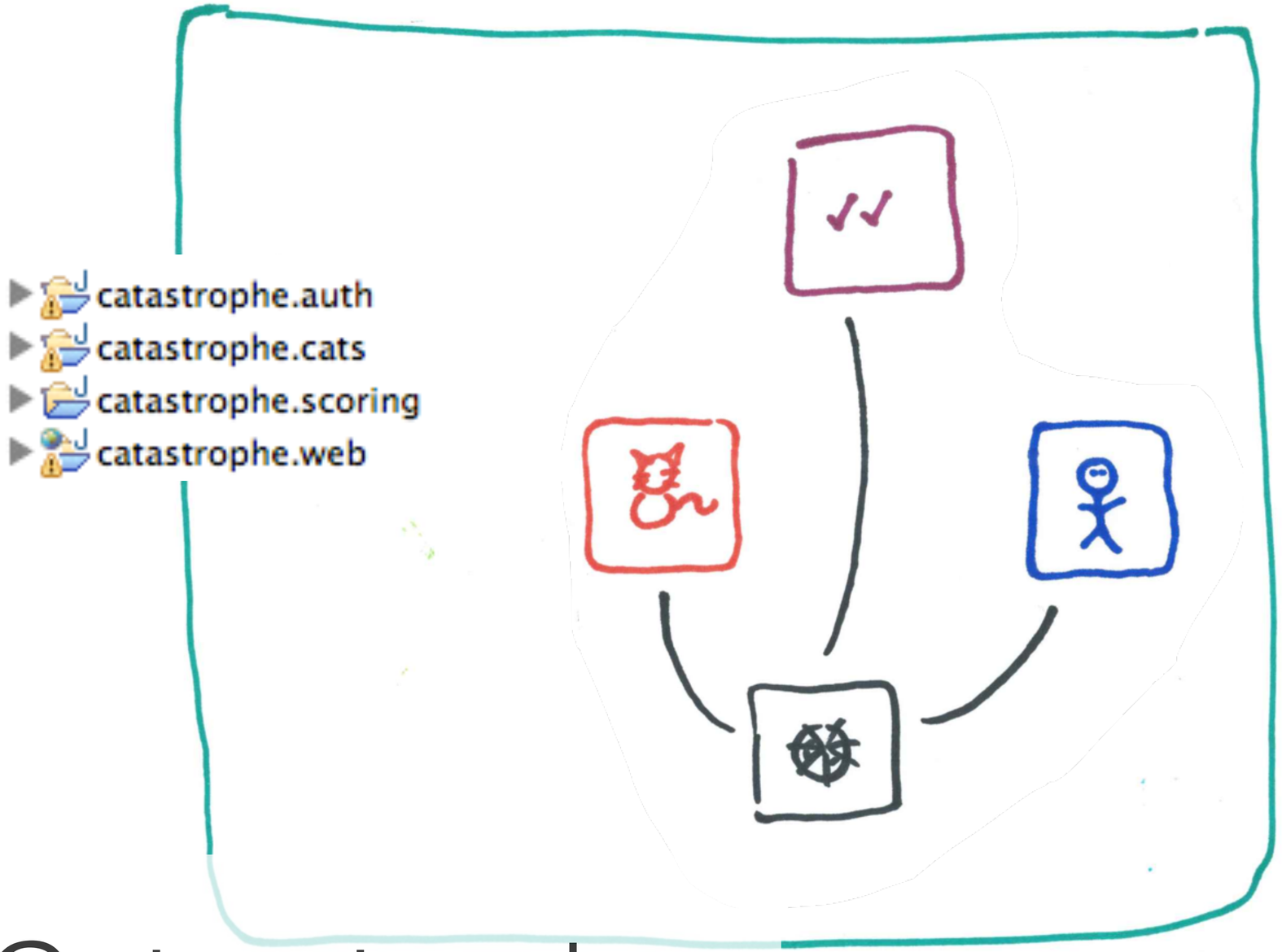
Monolithic application



Revolution

Monolithic application

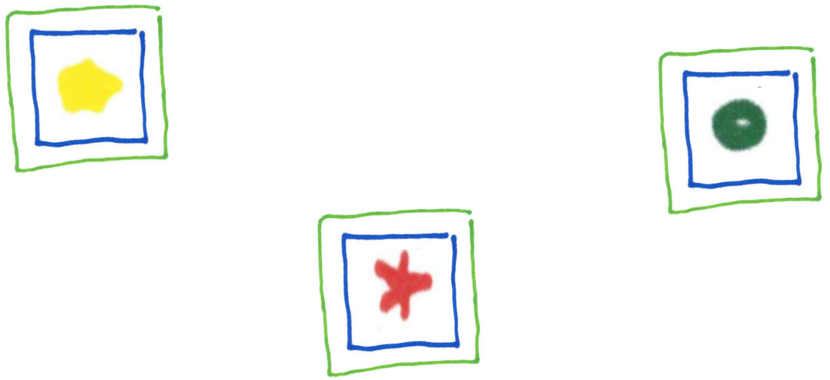
All good demos involve …



Cat-astrophe

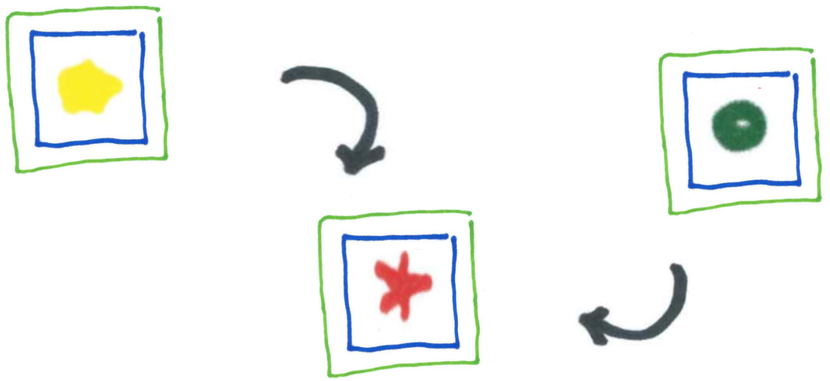
http://localhost:9090/catastrophe.monolith

Oops.



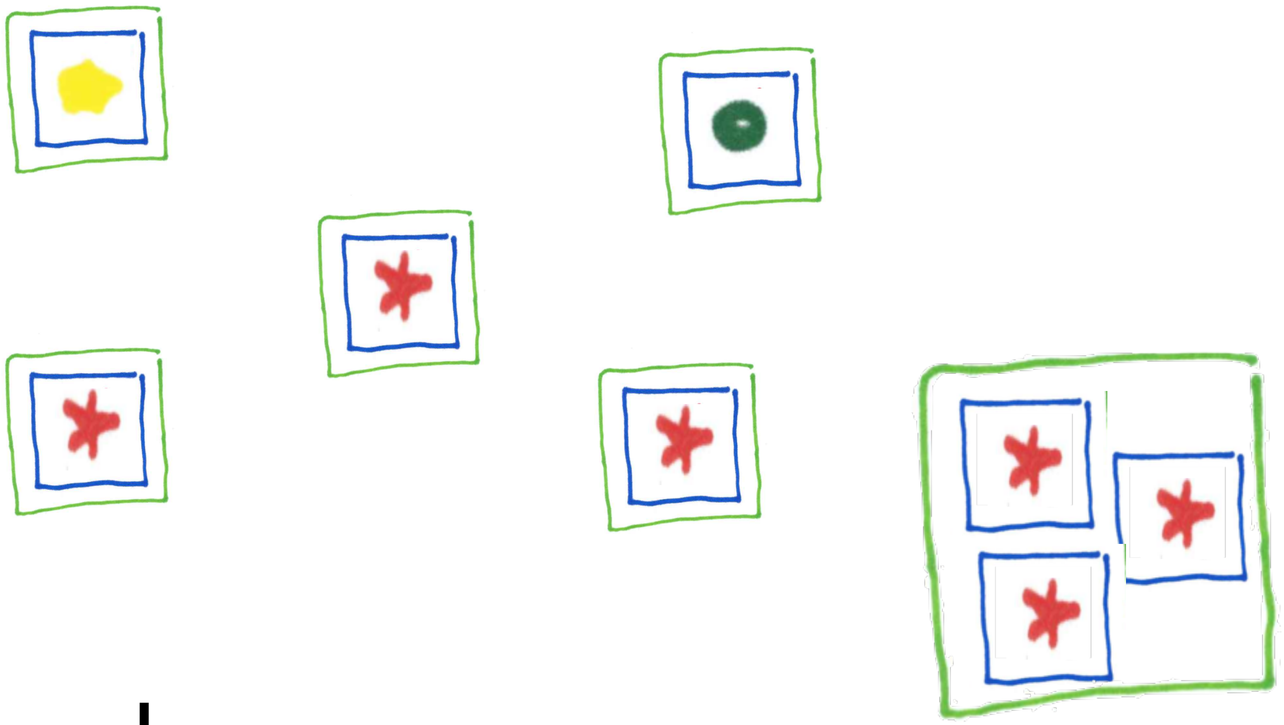
Modularity

Microservices application



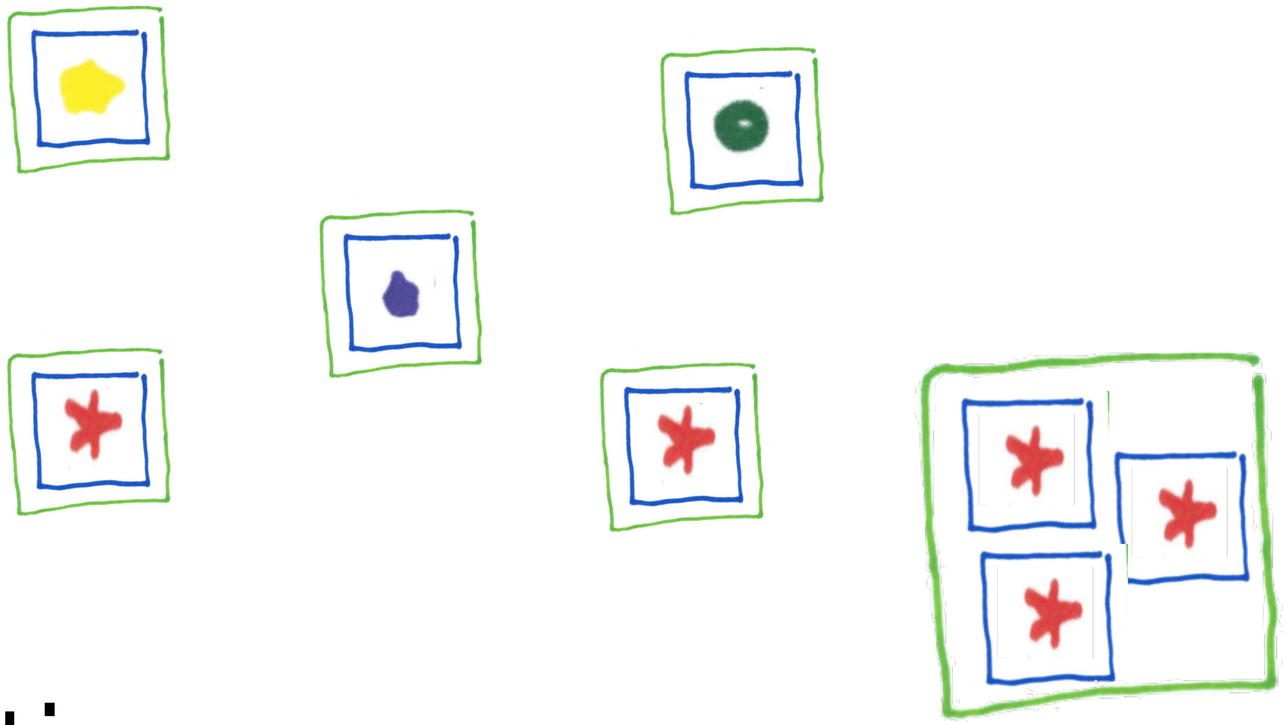
Interactions

Microservices application



Scaled

Microservices application

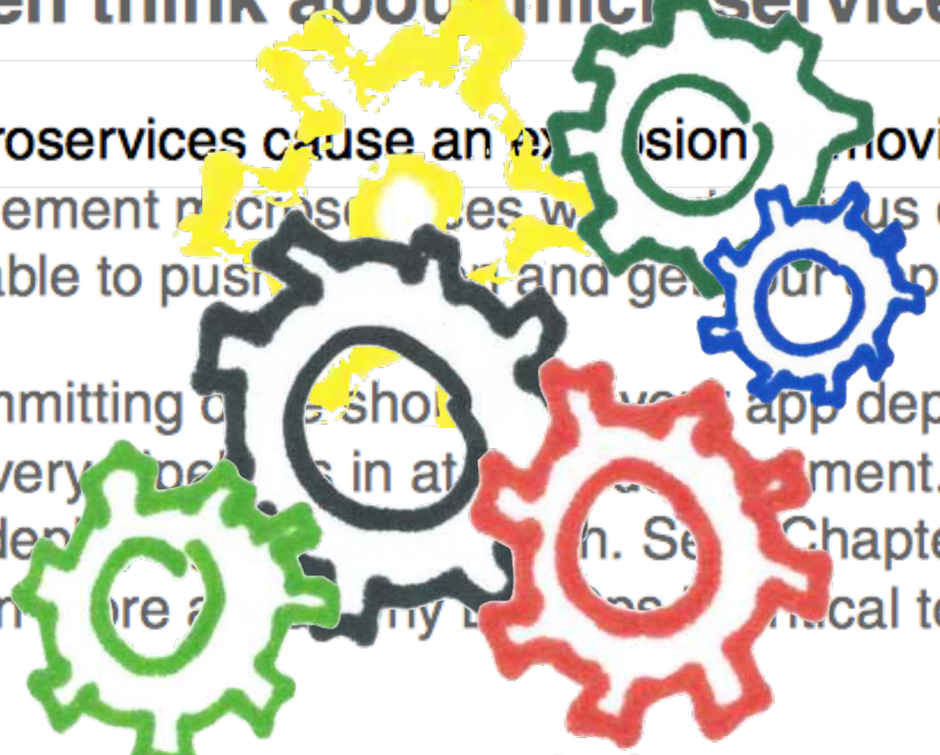


Evolution

Microservices application

It is important to be aware of when you are approaching monolith status and that occurs.

1.3.2 Don't even think about microservices without DevOps



Microservices cause an explosion in moving parts. It is not a good idea to attempt implement microservices without automated deployment and monitoring automation. You should be able to push a button—and get your app deployed. In fact, you should not even

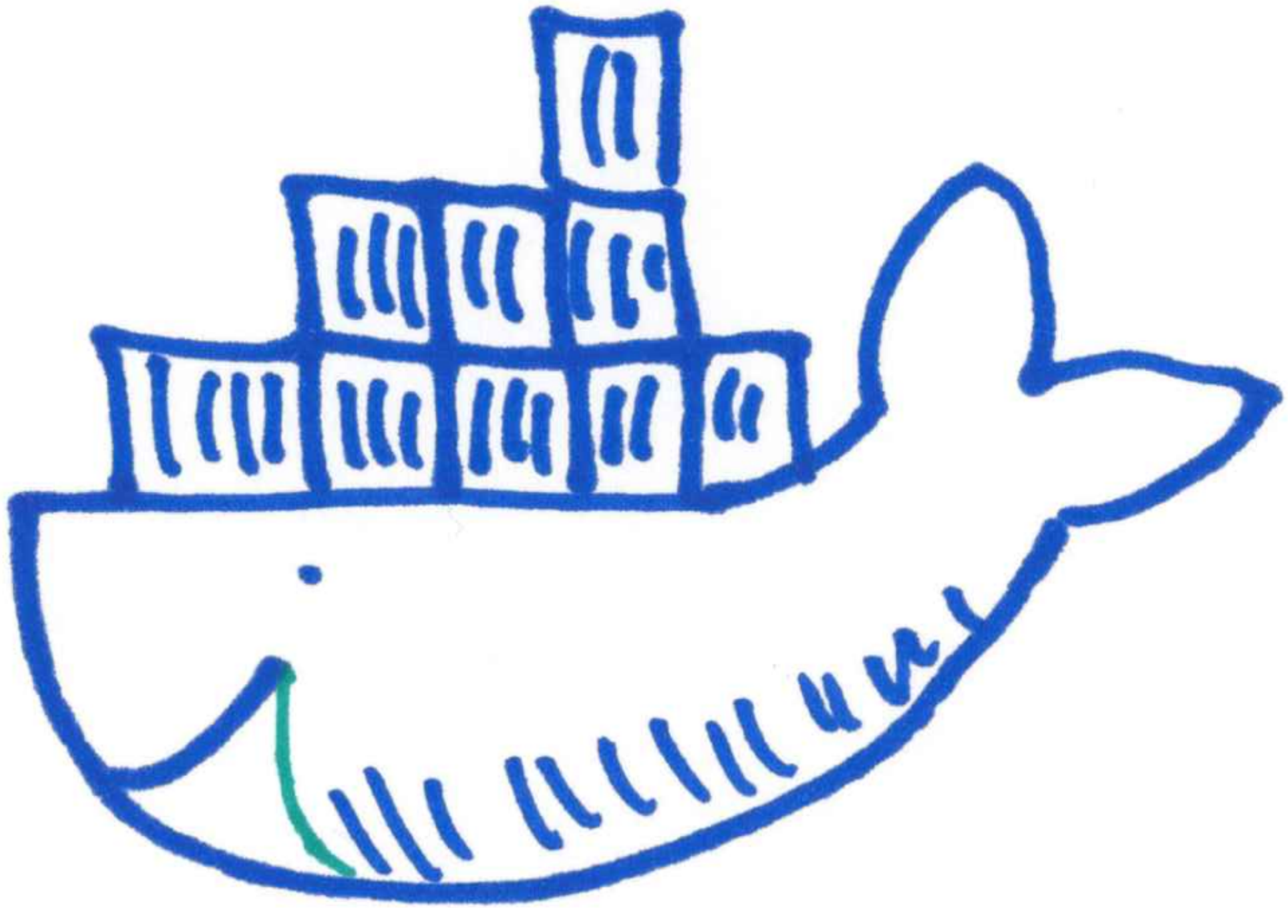
Committing code should get your app deployed through the commit hooks that trigger delivery pipelines in an environment. You still need some manual checks for deployment. See Chapter 3, "Microservices and DevOps" on how to learn more about why DevOps is critical to successful microservice deployment

1.3.3 Don't manage your own infrastructure

Microservices often introduce multiple databases, message brokers, data caching services that all need to be maintained, clustered, and kept in top shape. It really first attempt at microservices is free from such concerns. A PaaS, such as IBM


ANKER

Datacentre in a handbag



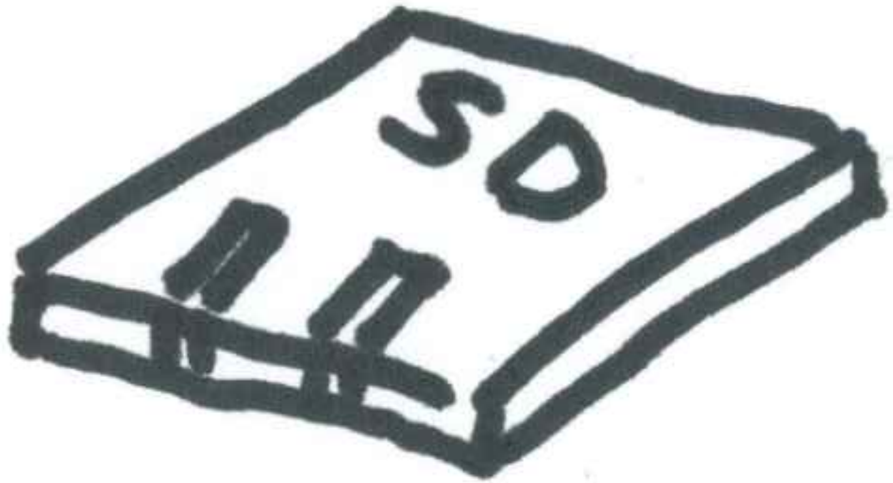
What, no Docker?

“Inconceivable!”

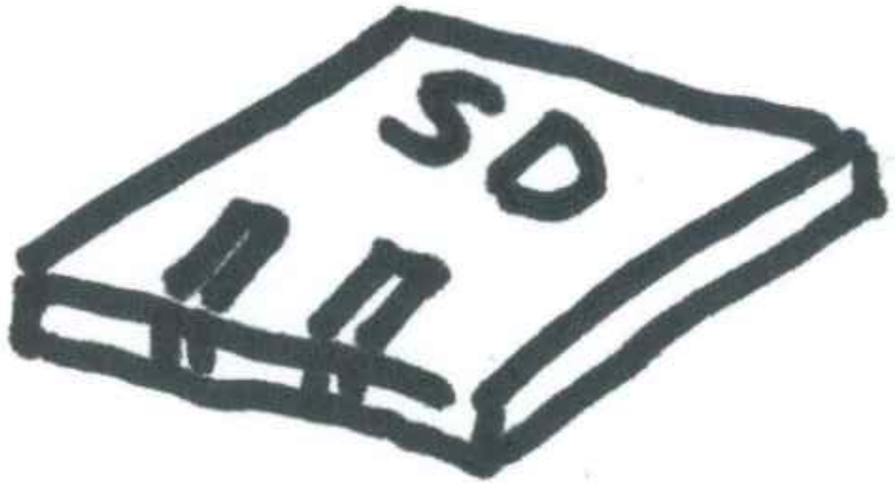
“You keep saying that. I dinna think it means what you think it means.”

Distributed computing fallacies

Complexity



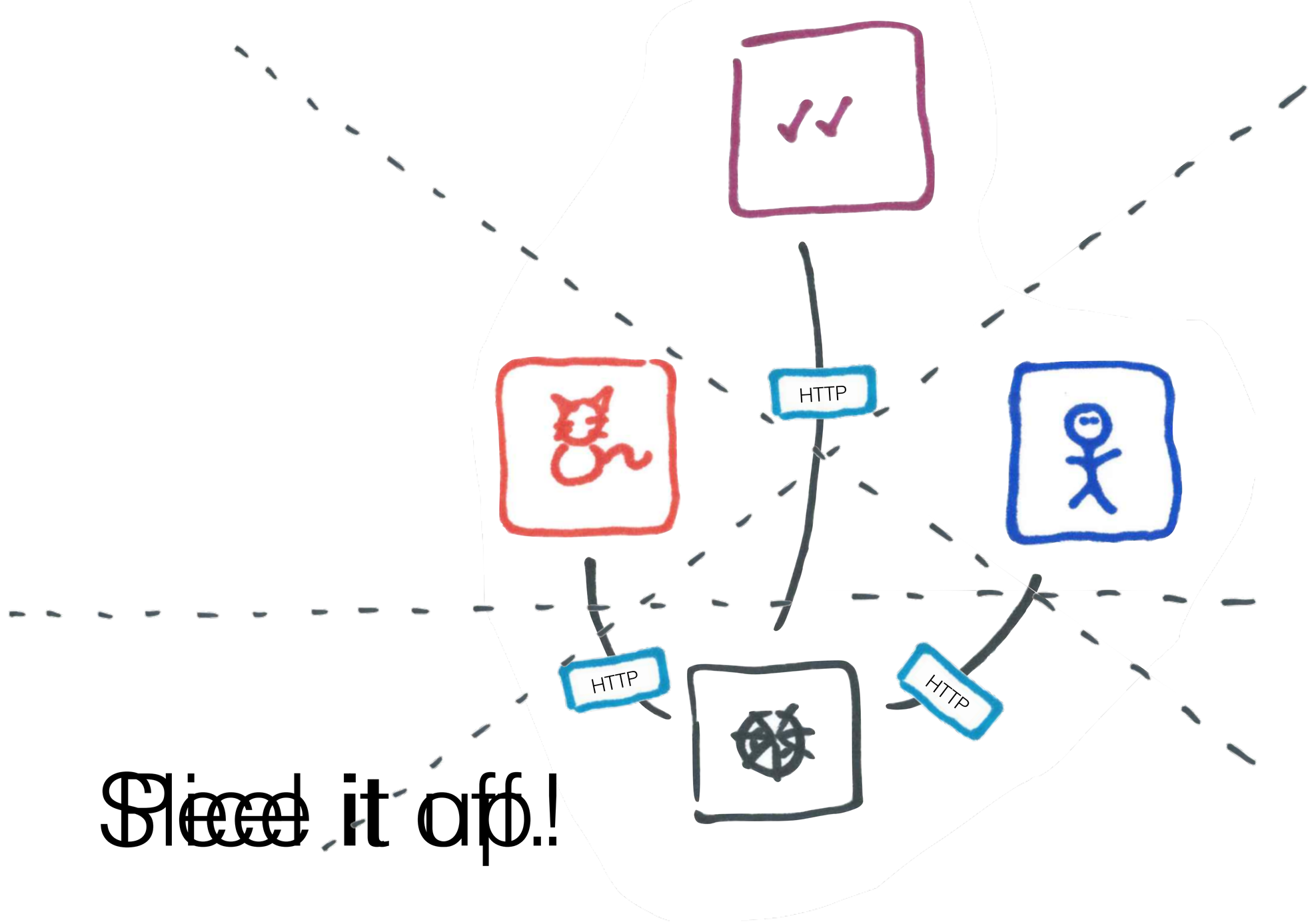
You need DevOps



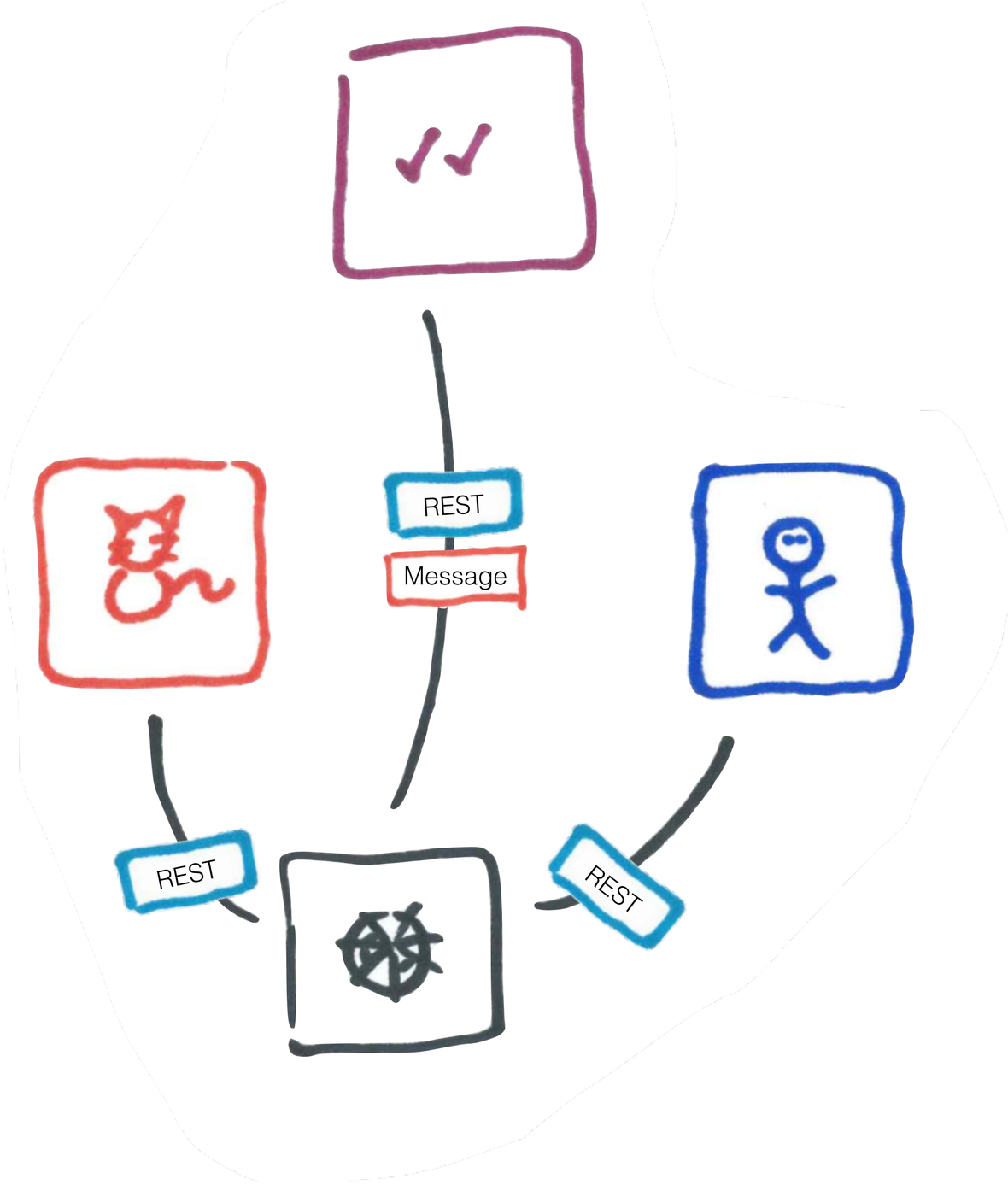
You need
100% automation

“…die Codebasis aufbrechen,
ohne sie zu zerbrechen…”

–Holly Cummins



Slice it off!


REST
Message
REST
REST

```
@ApplicationScoped
public class CatRepository {

  public Set<MiniCat> getAllCats() {
```

Exposing a service in a monolith

```
@Path("cat")
public class CatRepository {

  @Path("allcats")
  @Produces(MediaType.APPLICATION_JSON)
  @GET
  public Set<Cat> getAllCats() {
    …
```

JAXRS=magic

Exposing a microservice

```
@Inject
CatRepository catRepo;
  ...

Set<Cat> cats = catRepo.getAllCats();
```

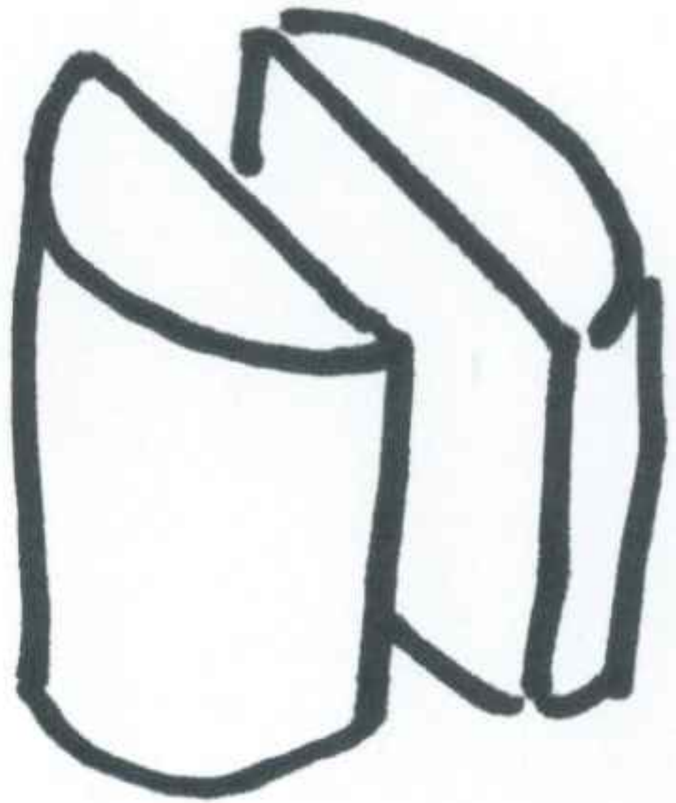
Consuming a service in a monolith

```
Client client = ClientBuilder.newClient();
WebTarget target = client.target(“http://localhost:9080")
      .path(“catastrophe.cats/rest/cat/cats");
Set<Cat> cats = target.request(MediaType.APPLICATION_JSON)
      .get(new GenericType<Set<Cat>>(Set.class));
```

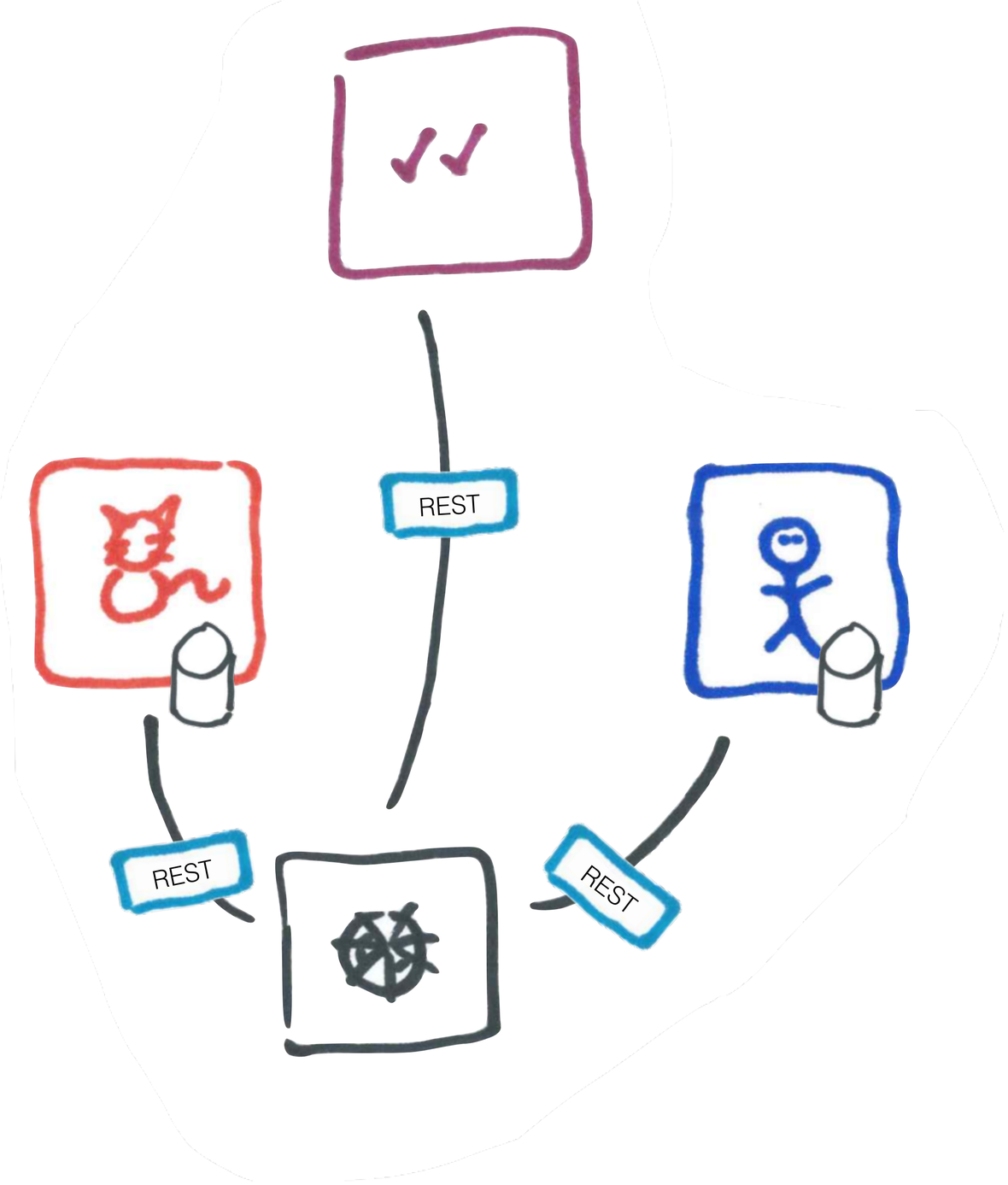
Consuming a microservice

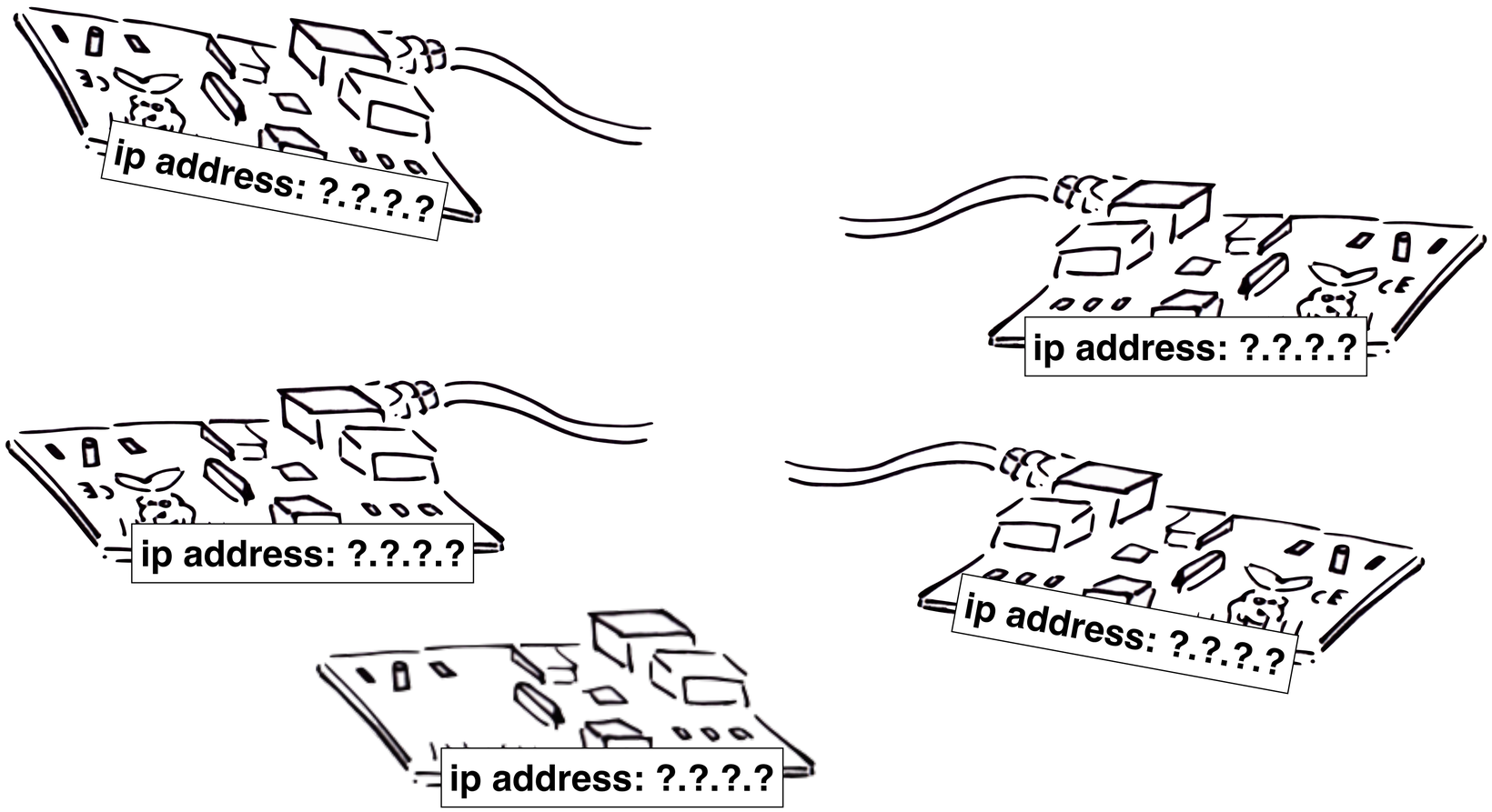
http://raspberrypiclearcase.local:9083/catastrophe.scoring.auth/rest/auth/leaderboard

Very nice.
Does it actually work?

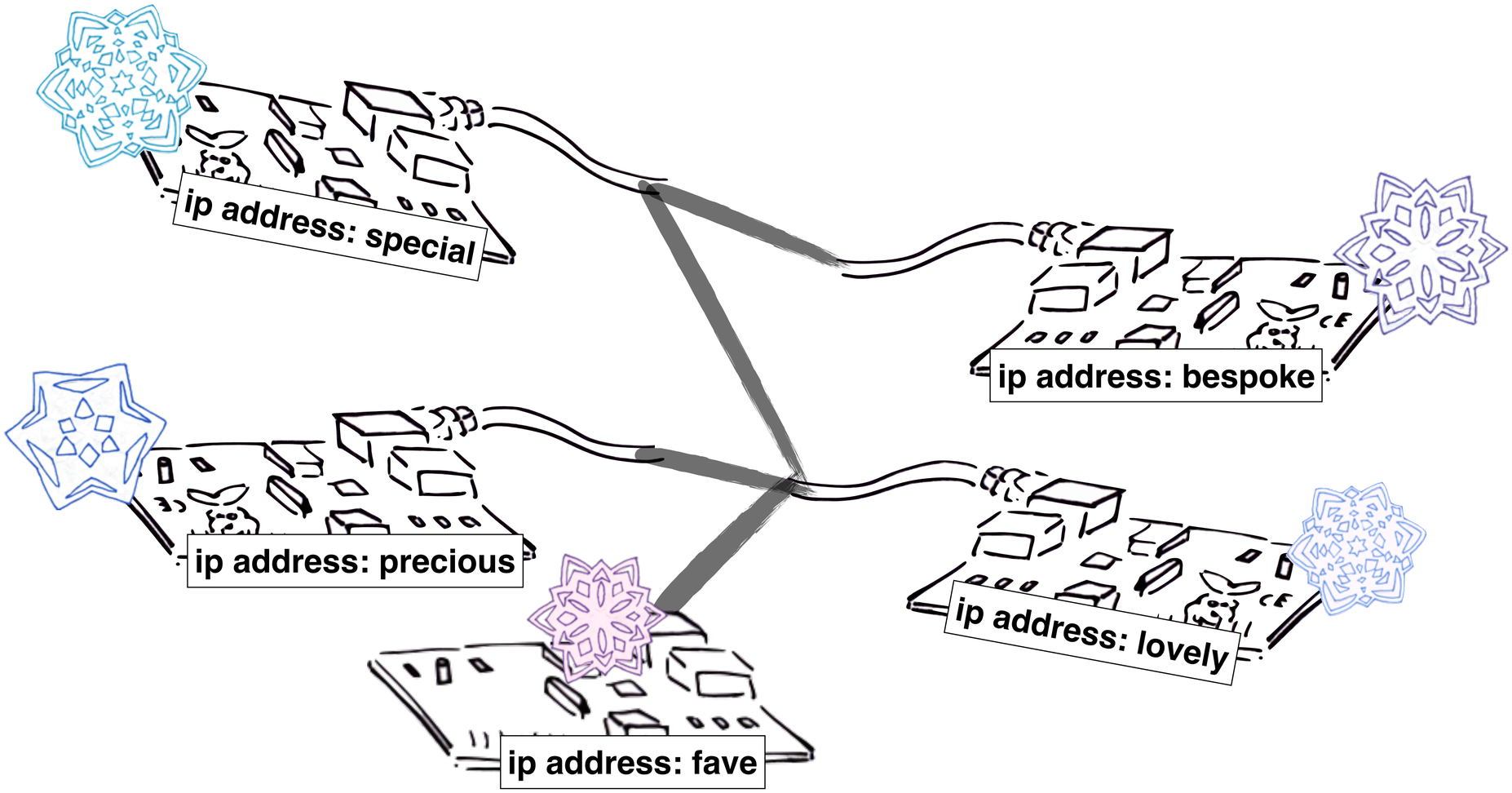


Don’t forget to slice
up the database too

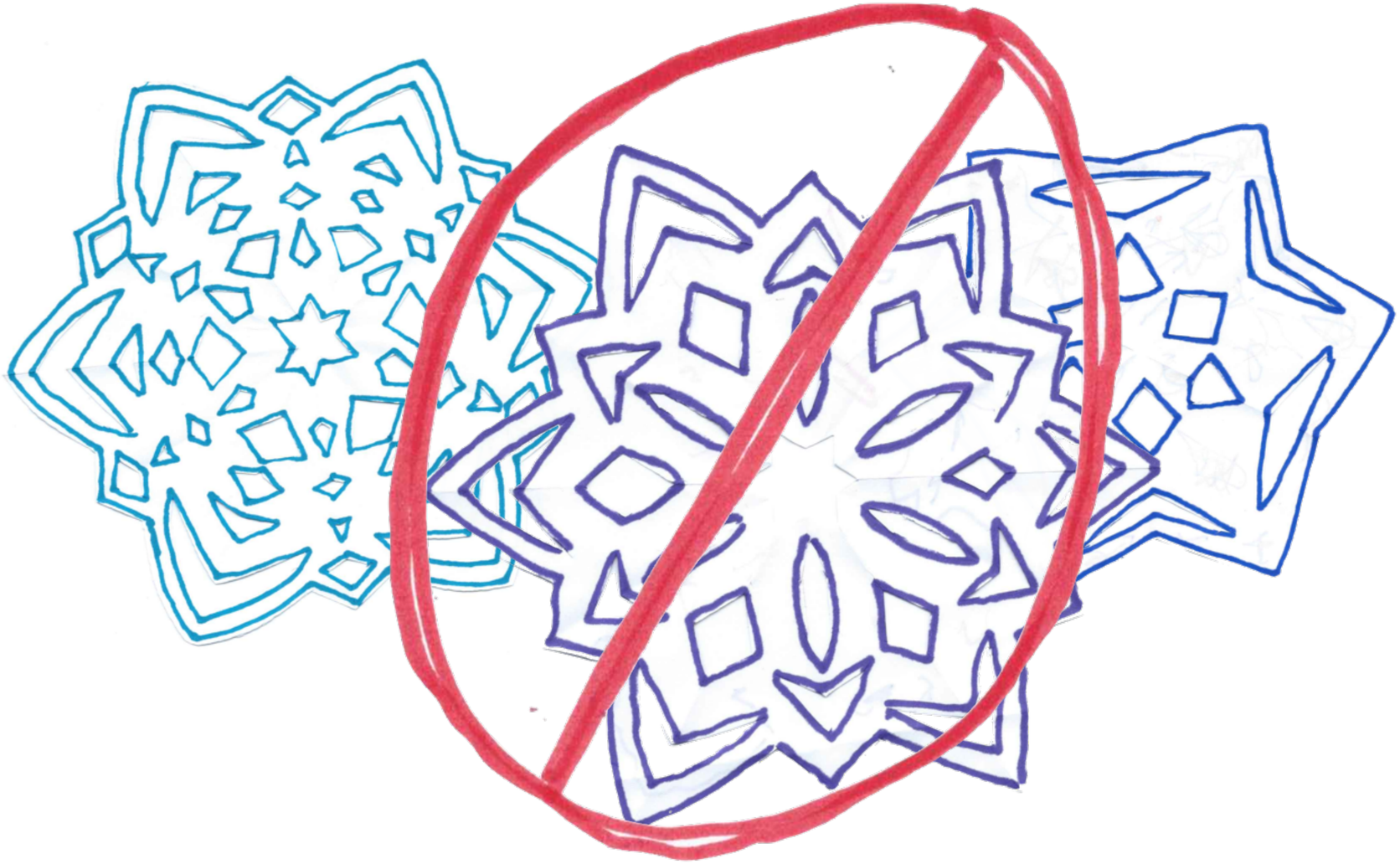

REST
REST
REST



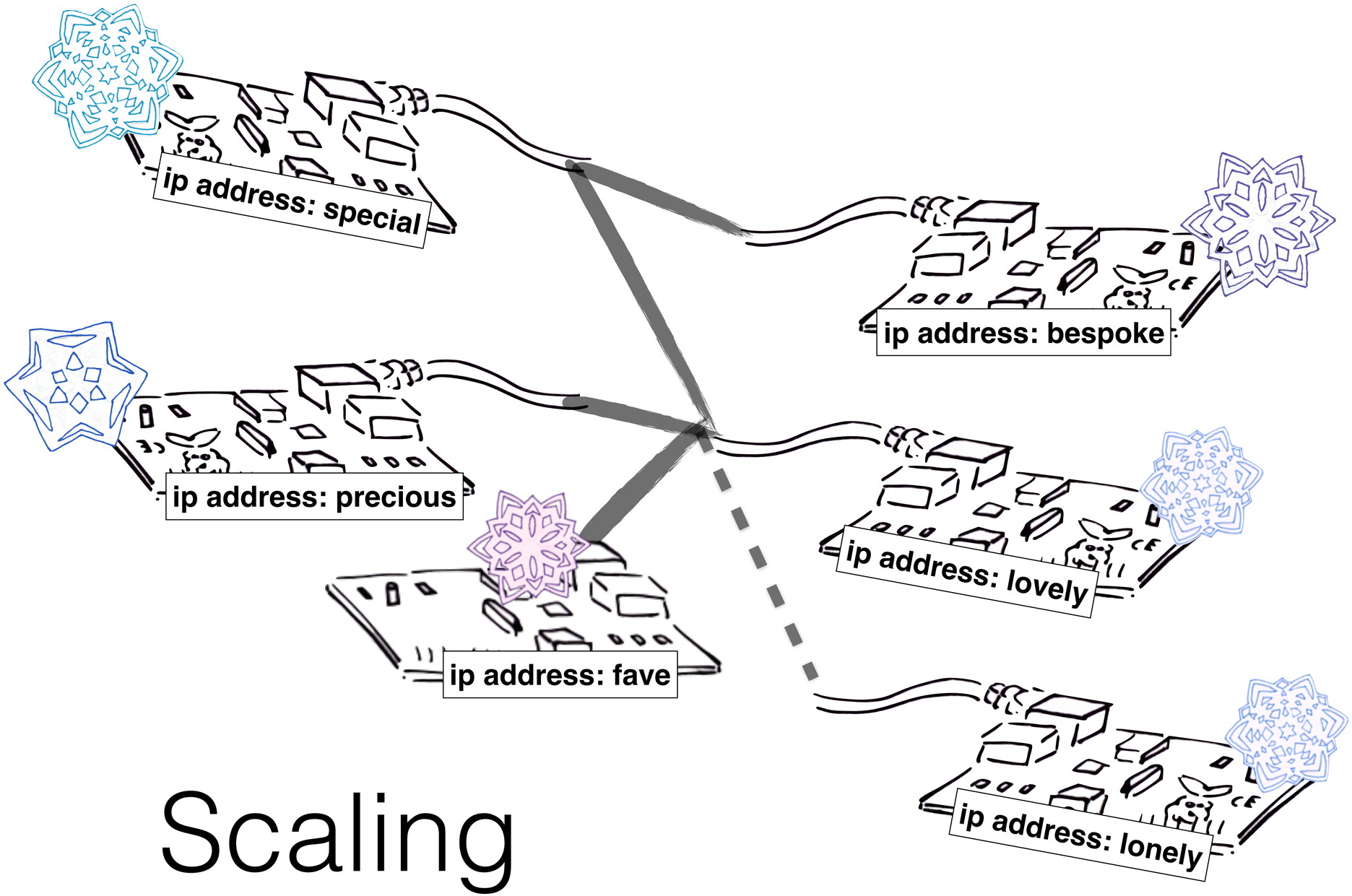
Network topology



Network topology



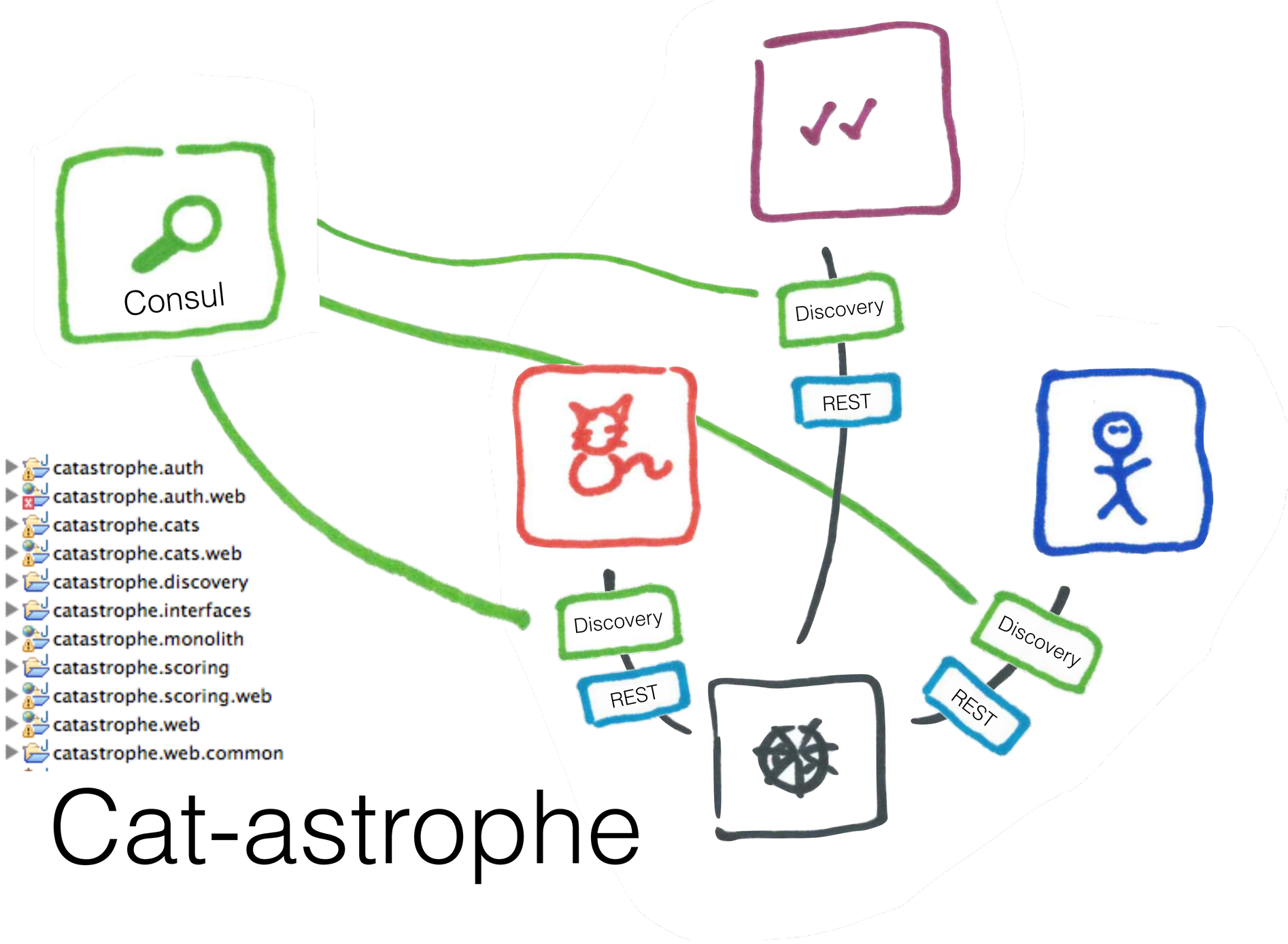
Disposability


ip address: special
ip address: bespoke
ip address: precious
ip address: lovely
ip address: fave
ip address: lonely

Scaling

- Kubernetes Docker
- Apache Zookeeper + Curator Java
- Eureka AWS SoftLayer
- etcd CoreOS
- Consul DNS HTTP Java

Service discovery



- catastrophe.auth
- catastrophe.auth.web
- catastrophe.cats
- catastrophe.cats.web
- catastrophe.discovery
- catastrophe.interfaces
- catastrophe.monolith
- catastrophe.scoring
- catastrophe.scoring.web
- catastrophe.web
- catastrophe.web.common

Cat-astrophe

Server configuration

```
<featureManager>
  <feature>jaxrs-1.0</feature>
  <feature>usr:discovery</feature>
    ...

<consul server="catastrophe.cat" />
```

Wouldn't this be nice?

WebSphere Liberty extensibility

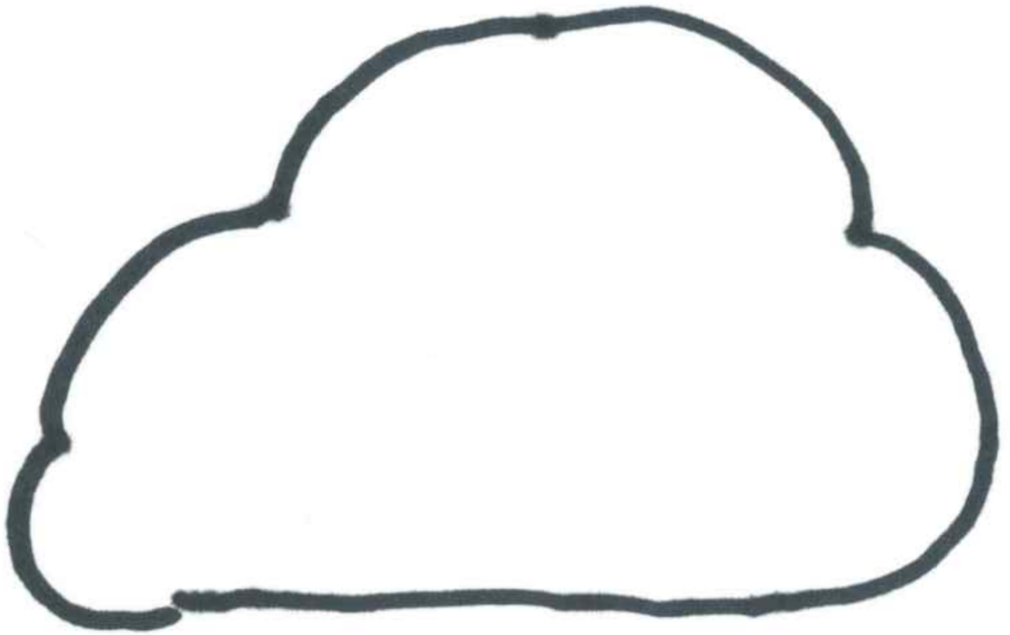
https://github.com/WASdev/sample.consulservicediscovery

http://consul.cat:8500

```
  public String getHostAndPort(String serviceName) {

        List<CatalogService> services = client.getCatalogService(serviceName,
QUERY_PARAMS).getValue();
        int numServices = services.size();
        if (numServices > 0) {
          // Do a simple random-robin :)
          int index = RANDOM.nextInt(numServices);
          CatalogService service = services.get(index);
          return service.getServiceAddress() + ":"
               + service.getServicePort();
        } else {
          System.out.println("No services available with name "
               + serviceName);
        }
```

http://raspberrypi.local:9080/catastrophe.web



https://console.eu-gb.bluemix.net

http://catastropheweb.eu-gb.mybluemix.net/catastrophe.web/

Are we done?

Any questions?

bluemix.net
@holly_cummins