10 November 2020

SDE Project



A technological guide for the underlying distributed design of modern systems.

Note: This guide is at the conceptual level. It requires generalized knowledge of all information technology above the hardware level to be fully utilized. It does not contain step by step guides to how to set up any of the technologies mentioned.

# Systems Design in a nutshell

## Introduction

System design choices should first be influenced by the goals of the system. Different systems have different tolerances and strengths. A bank system is designed for reliability security and consistency while a pizza chain’s system is designed for speed, efficiency and cost reduction. Consistency and security are less of a factor in a pizza chain, a wrong order is not going to cause potentially catastrophic results and a delivery man or pizza shop being robbed is not going to cause the entire chain to go bankrupt (if managed properly).

There is more than likely a software equivalent of any logistical paradigm in existence. A good way of thinking about it is for any function a human does in an organization, there will be a software equivalent doing the same function in a system.

## Conceptual & Technological design

Designing a system on a conceptual level and on a technological level are completely separate activities, although they are related, they are not equivalent.

### Conceptual

Conceptual design choices are relatively straight forward, we decide what the system is doing and what it should and should not be good at while doing it.

Conceptual choices are limited by and influence technological choices.

### Technological

Technological design choices are influenced mainly by 3 factors.

1. Scale

How many people are going to use the system?

How much data will be moving around the system?

What is the geographical scope of the system?

1. Medium

How does the system interact with humans? Or does it even need to?

1. Conceptual requirements

What are the features the system is required to have?

Just to add another level of difficulty. The technological choices made should ideally be able to support expansions to the systems that have not yet been thought of or are not yet possible as well as shifts in conceptual features and requirements

# Distributed Software Architecture

Conceptually most system designs are a copy of logistic supply chains we have already figured out in the physical world. We will take a look into one of these analogical examples of a distributed architecture and also show their software equivalents.

This example describes what is known as micro service architecture. Do note that while micro services are great, people tend to go overboard and fragment too many things into too many services. There is no set guide or method on how to avoid this, it is pretty much up to the development team to be intelligent enough to design it well.

The microservice philosophy is essentially capitalism in software form. We want each citizen to do one thing extremely well.

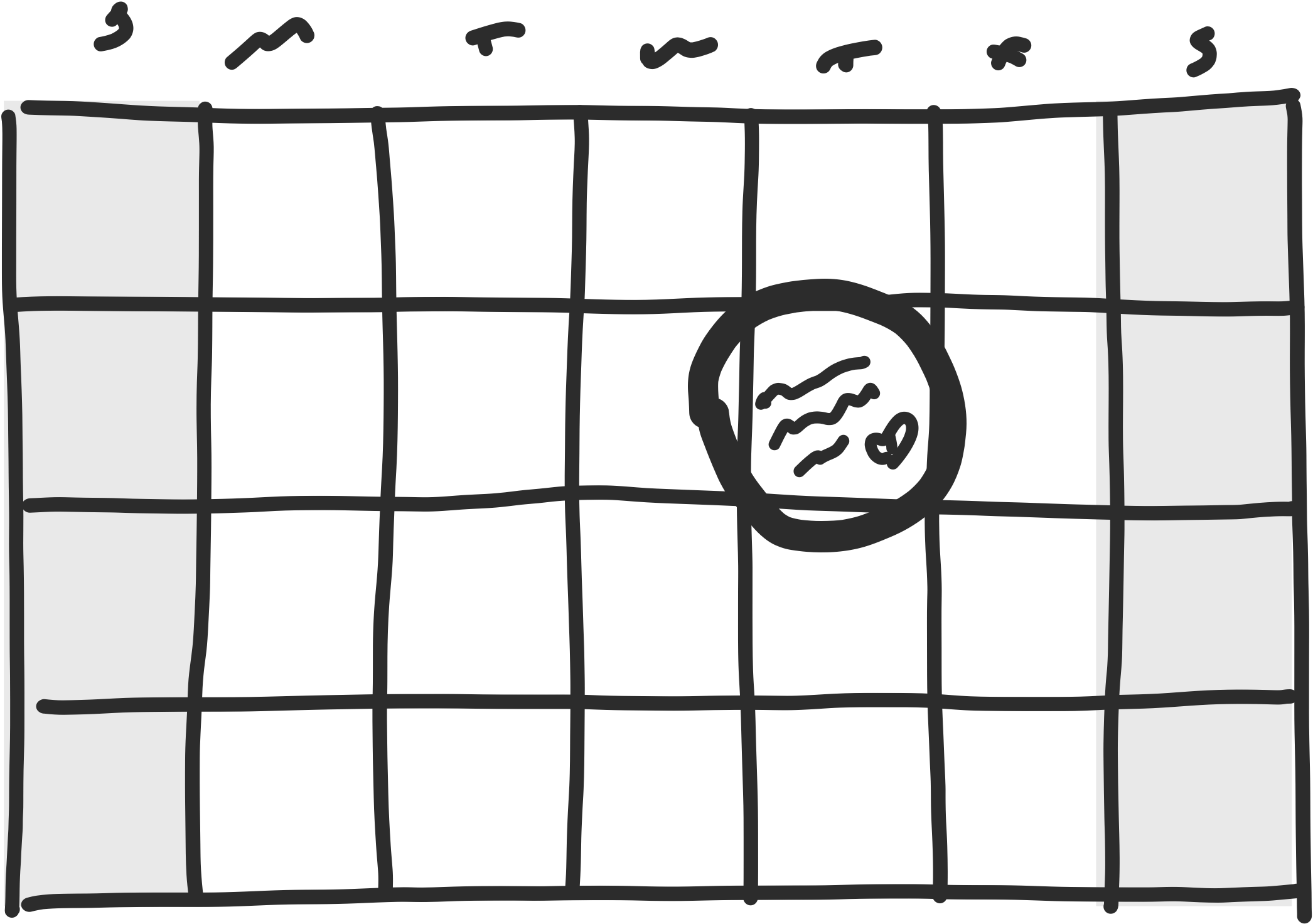
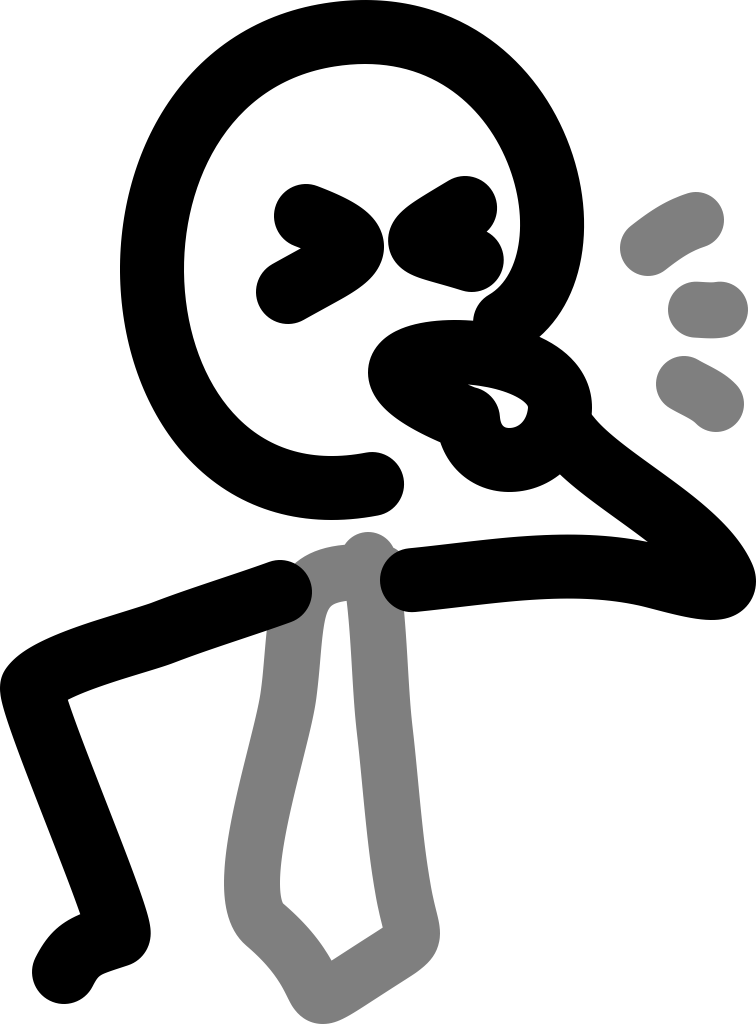
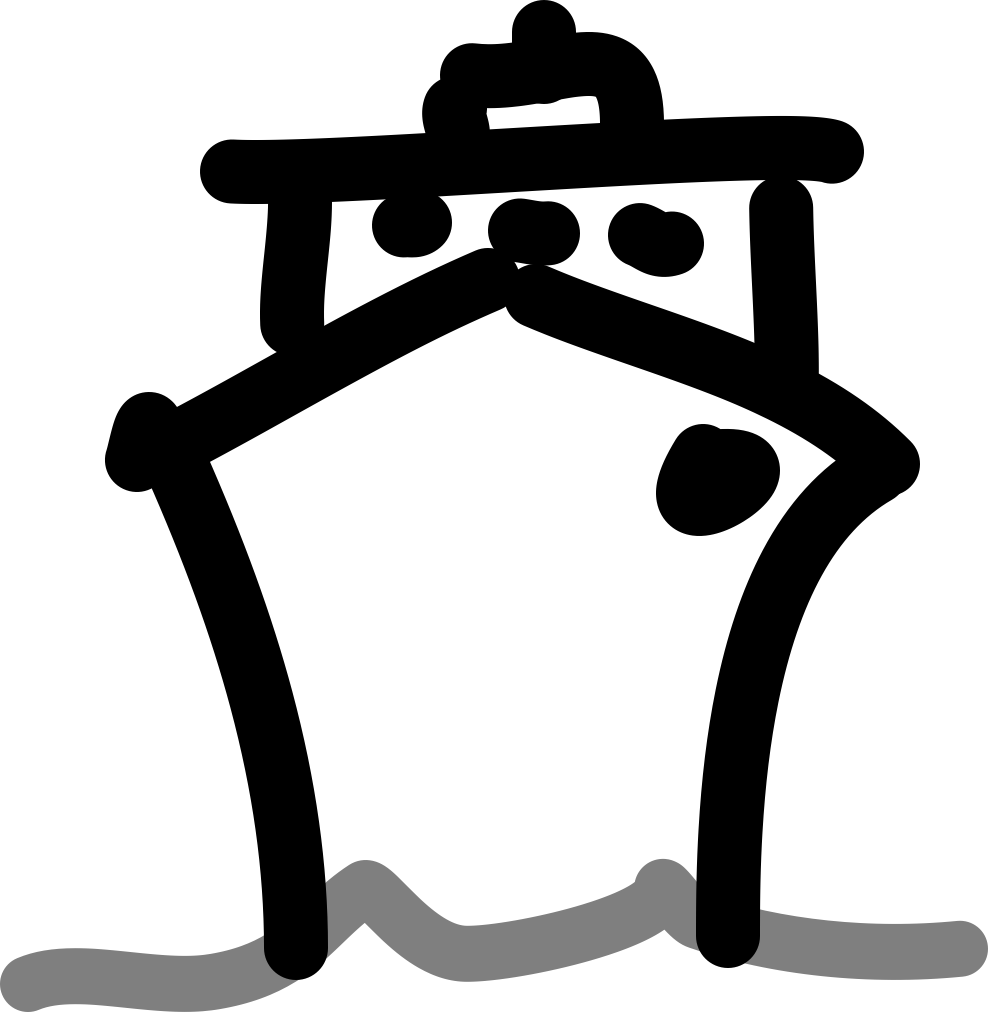
### Pizza Restaurant

There can be many of each component in the following diagram.

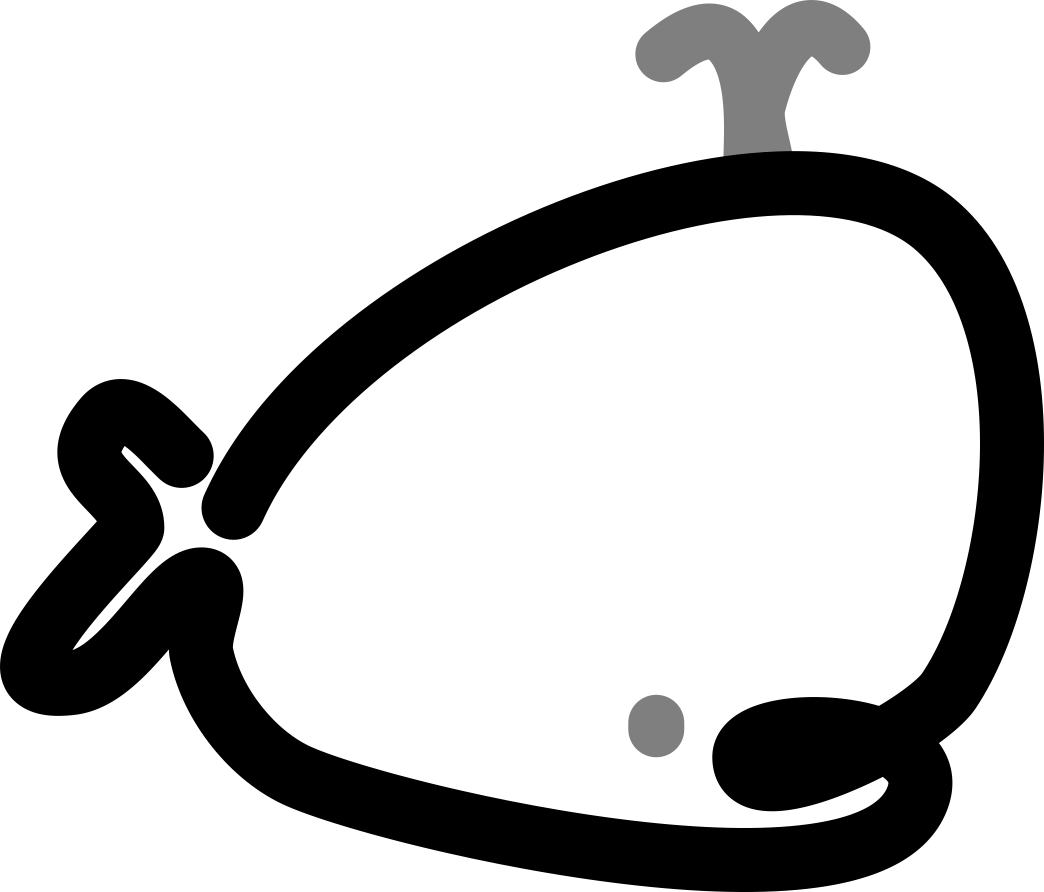
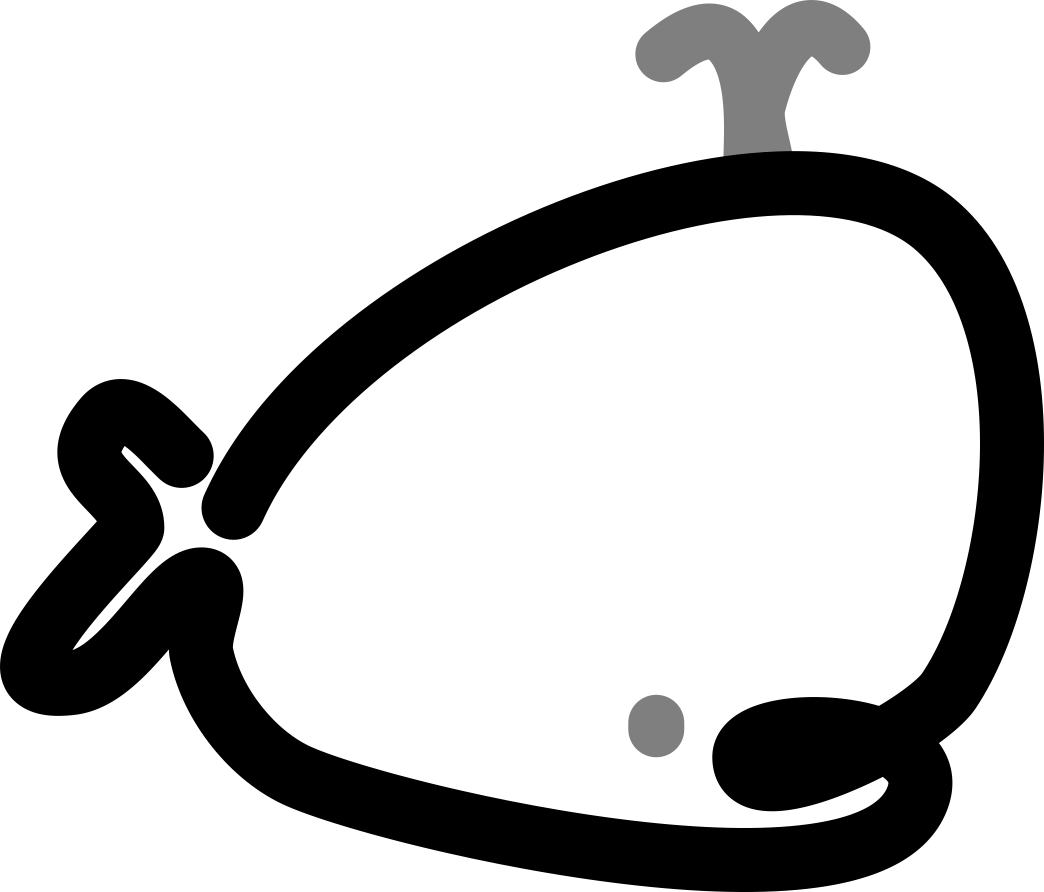
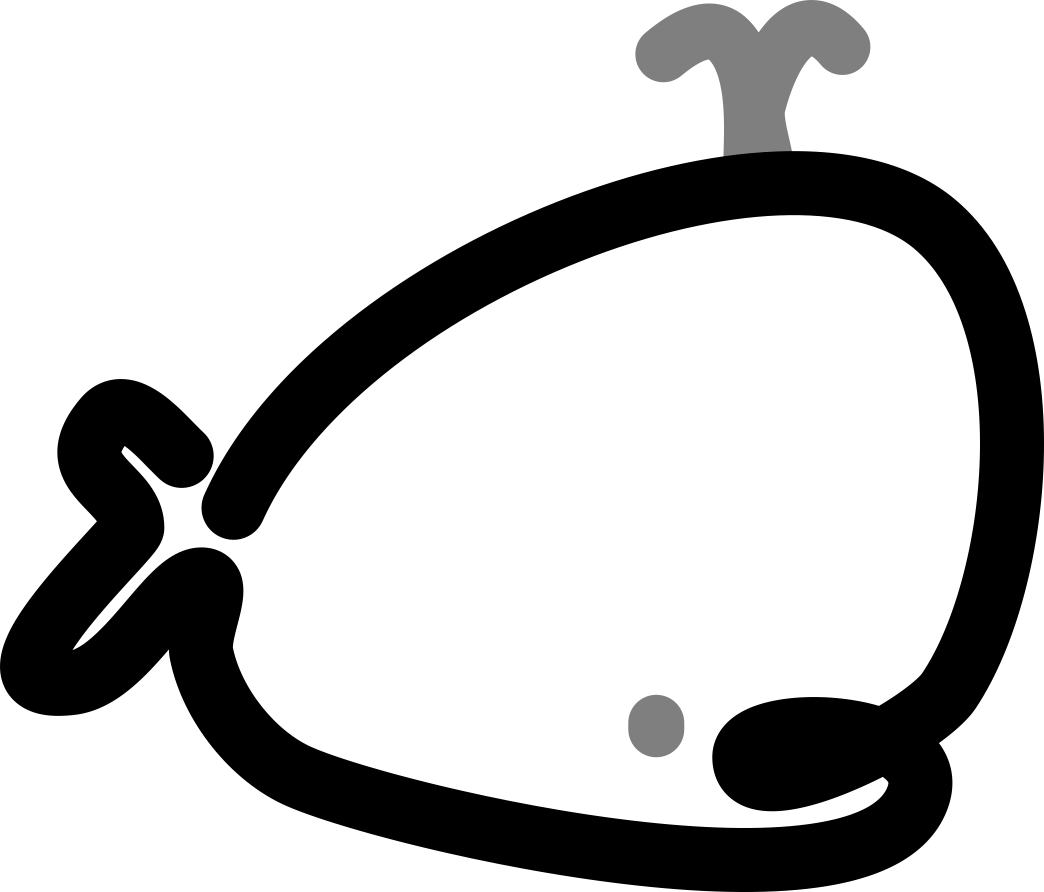
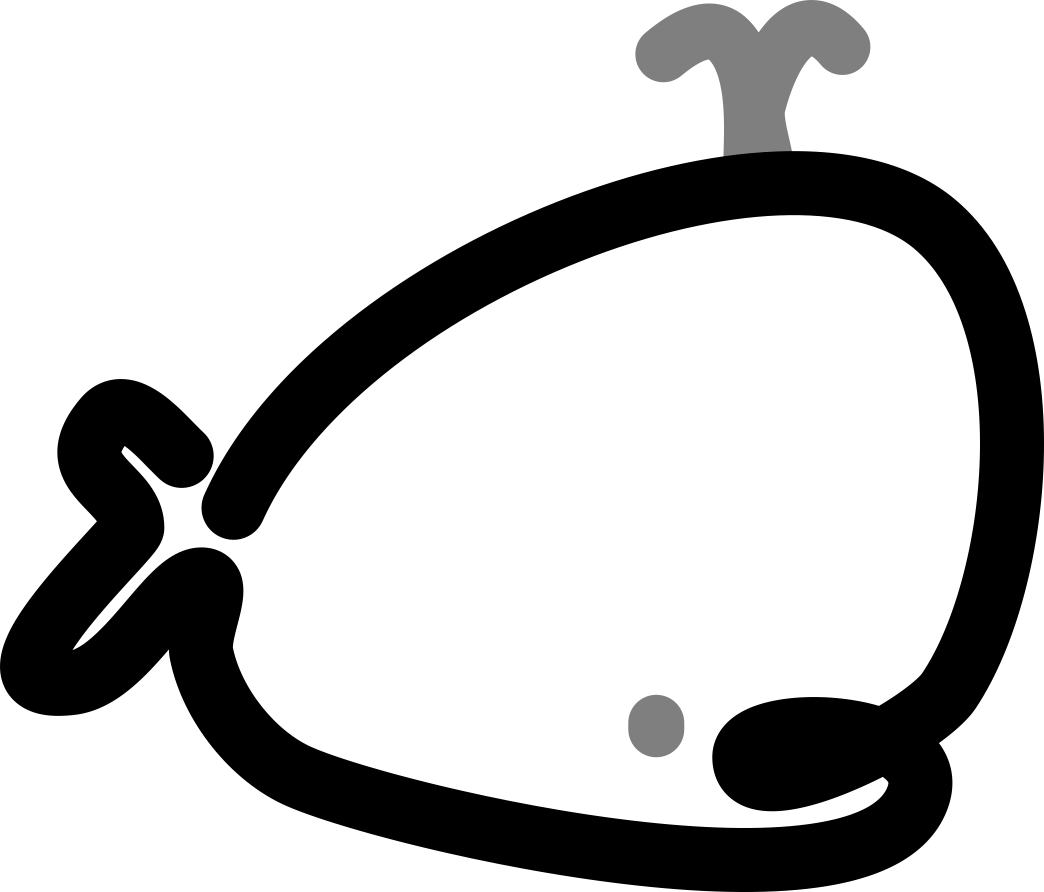
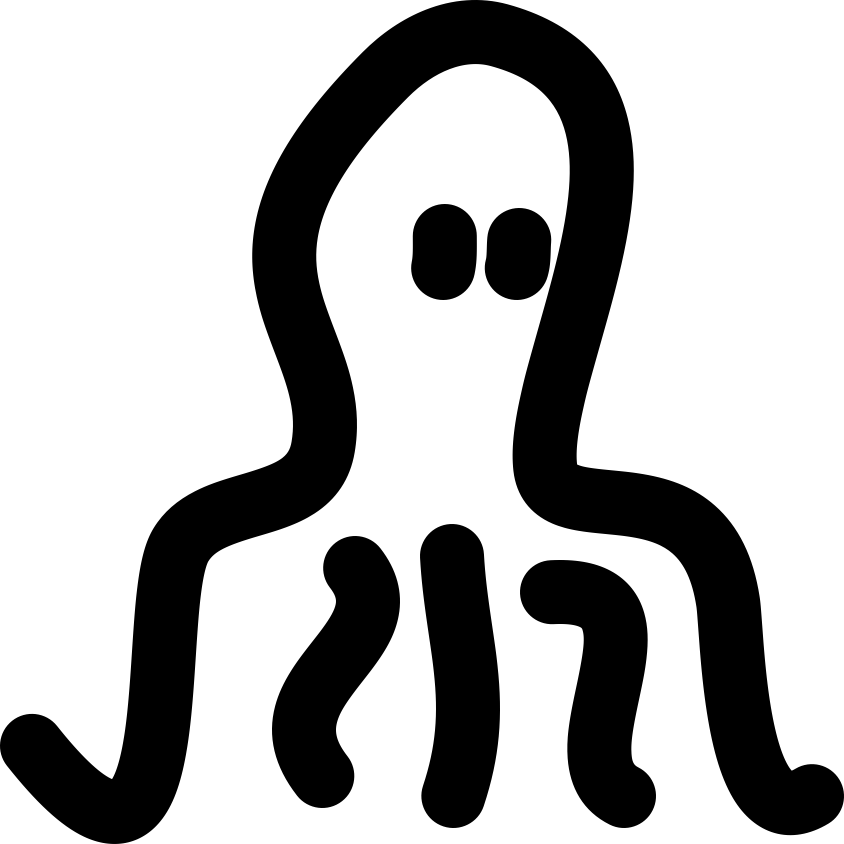
We will first look inside a single chain restaurant.

Note: We will look into each label technology further into the guide.





Kubernetes



Argo

Docker

Docker

Docker

Docker

A picture containing shape

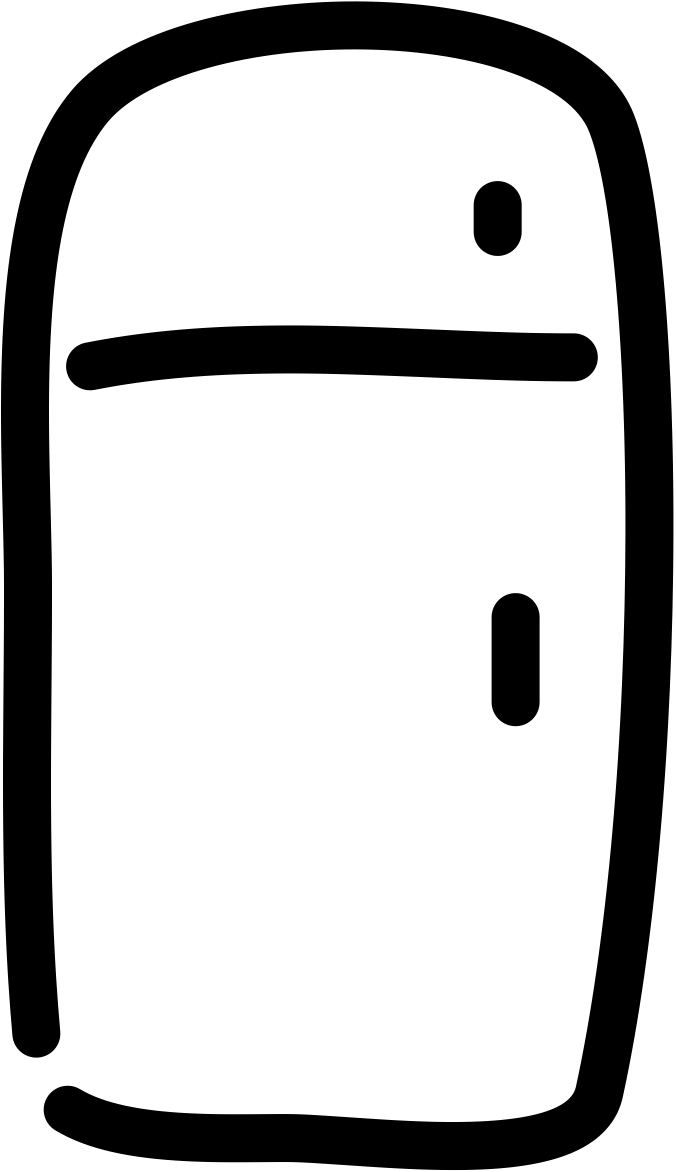
Description automatically generatedA picture containing shape

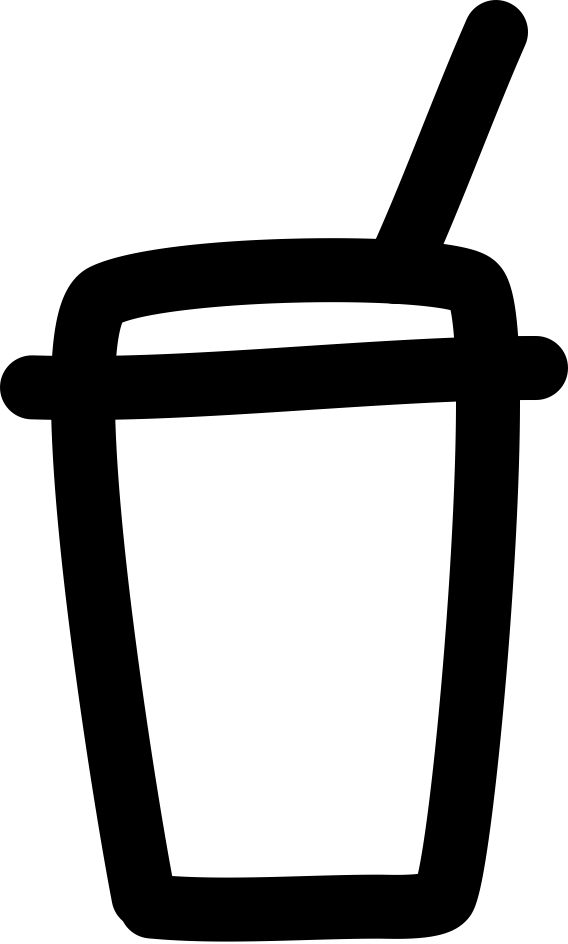
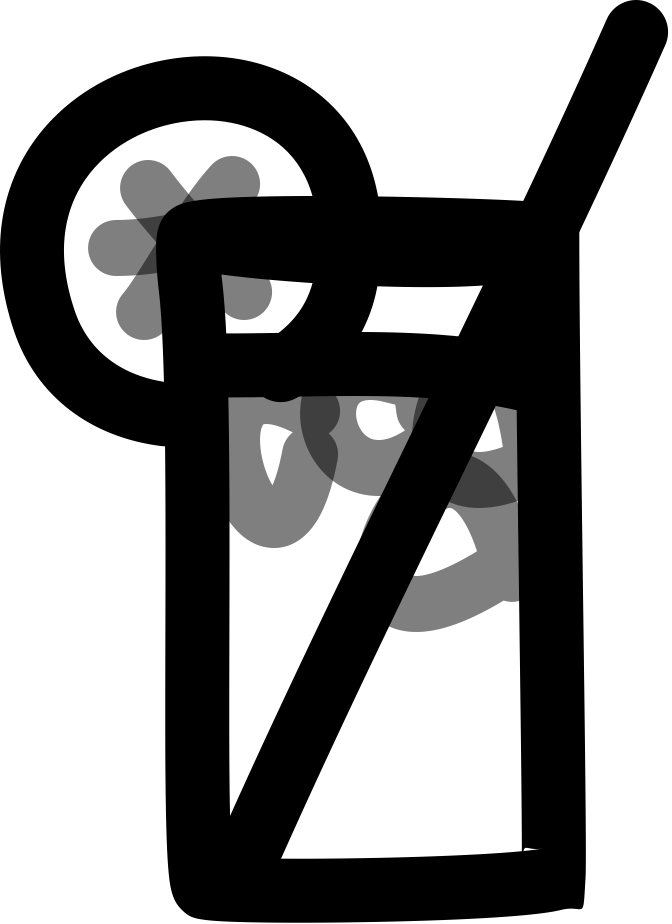
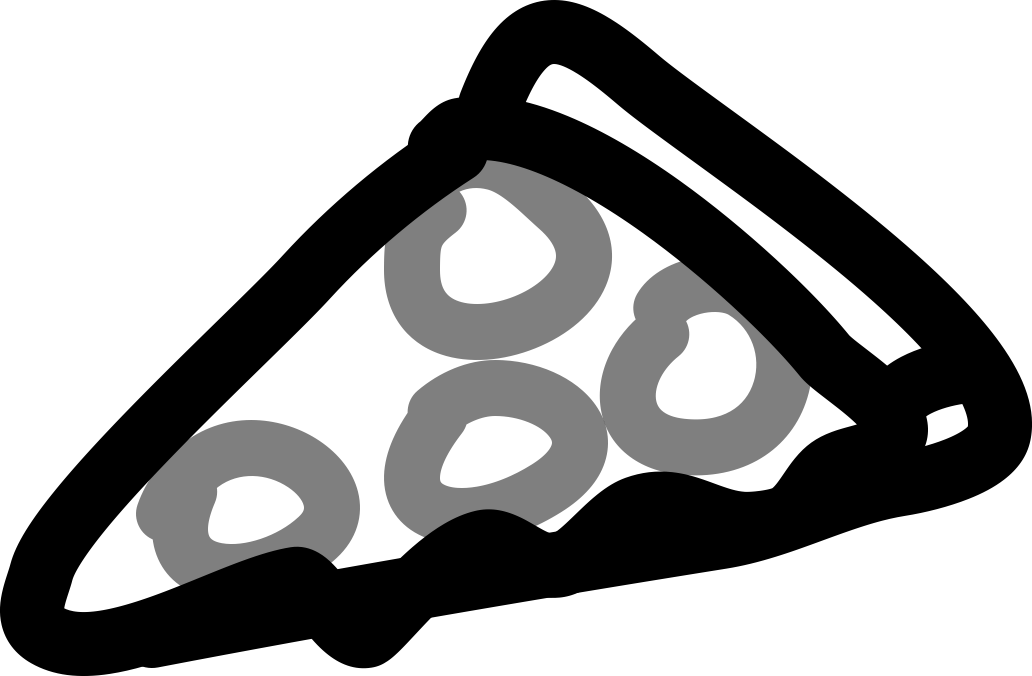
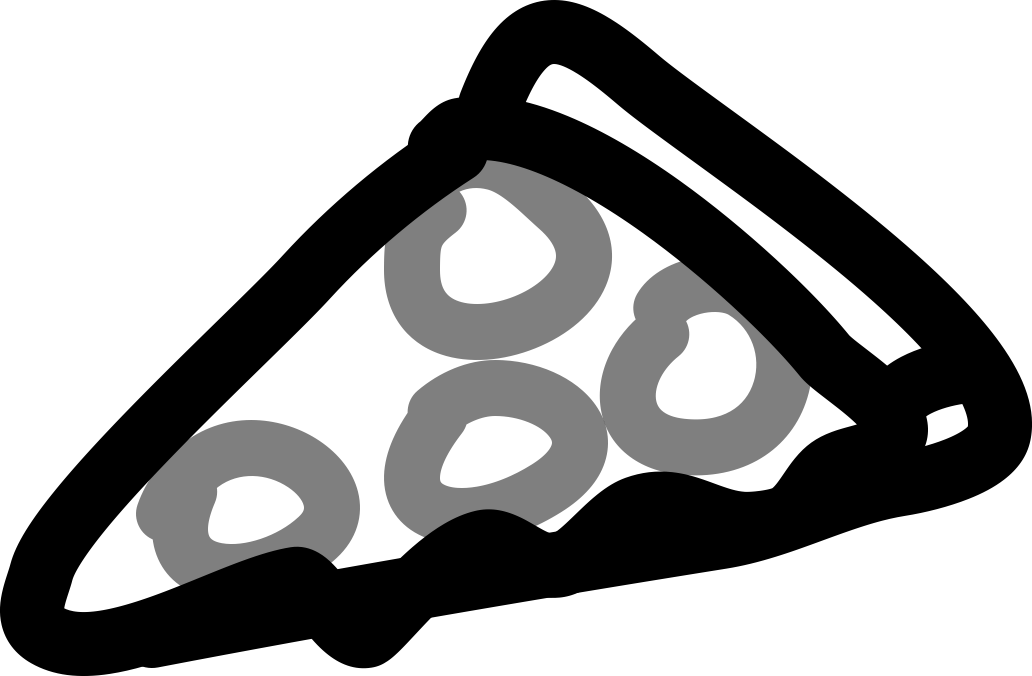
Description automatically generatedA picture containing shape

Description automatically generatedA picture containing shape

Description automatically generatedA picture containing shape

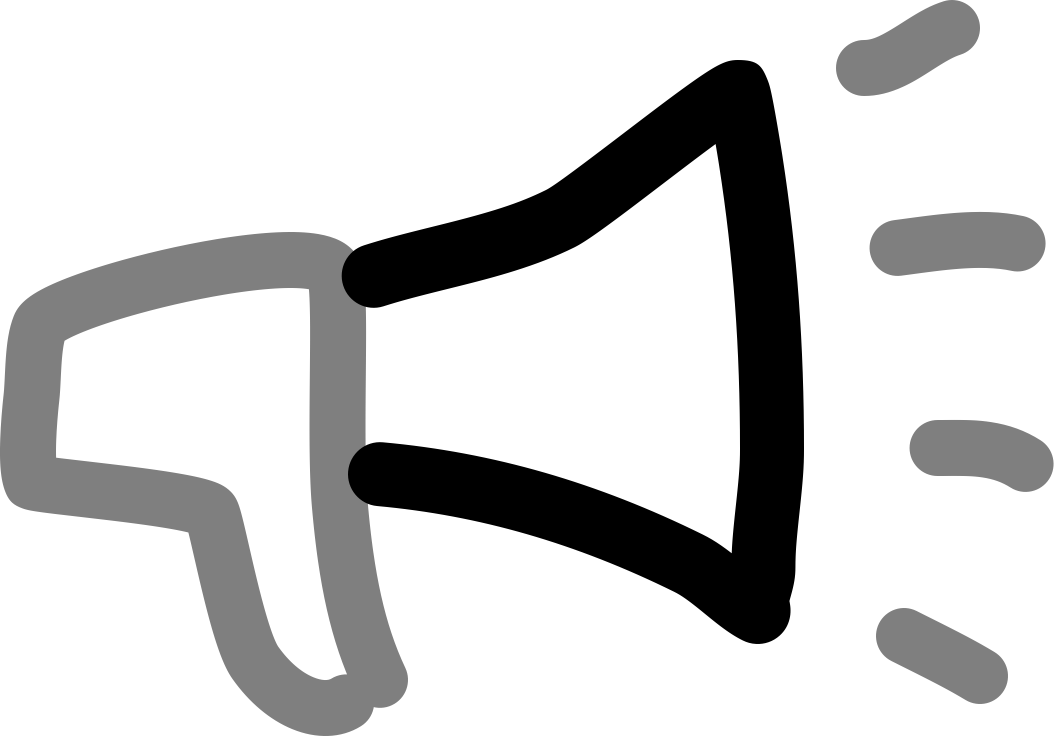
Description automatically generated





Order

Communication/interactions between containers are handled by message queues, RPCs, APIs, sockets, etc.



* Services, Specialized Programs

Each worker specializes in a specific role in the restaurant.

* Container Orchestration, Kubernetes

The manager of the restaurant keeps track of everyone working in the restaurant and schedules work hours according to busy hours and employee availability.

* Workflow Orchestration, Argo

The cashier takes orders and relays them to the chefs, including different requirements for each order. Eg. no cheese, ham and tomatoes.