Software Engineering for AI-Enabled Systems

SOFTWARE
SYSTEME

UNIVERSITÄT
LEIPZIG

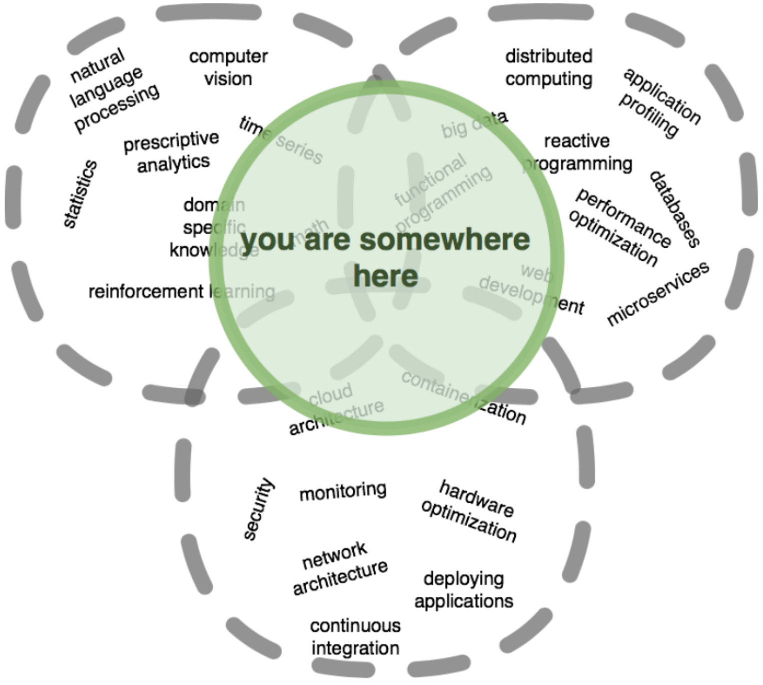
Prof. Dr.-Ing. Norbert Siegmund
Software Systems

Topic I:

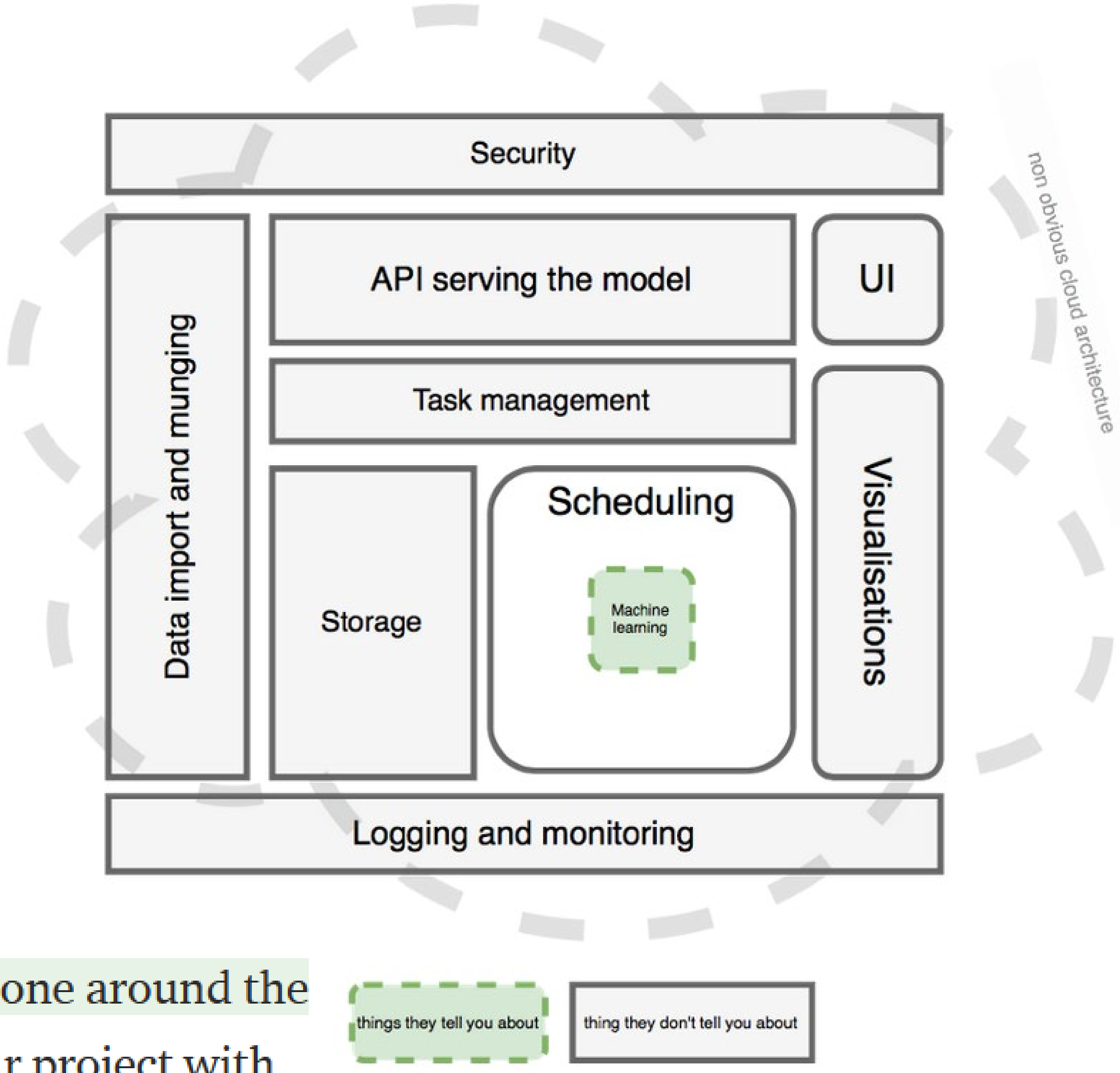
Basics of Software Engineering

Why SE Basics are Needed?

In reality **the majority of your code is not tied to machine learning. In fact, the code regarding it usually takes just a few percents of your entire codebase!** Your pretrained black box gives only the tiny JSON answer — there



not even a quarter of all the buzzwords available is shown



that was not on Coursera, was it?

No matter how small the problem is, the amount of work to be done around the machine learning itself is tremendous, even if you bootstrap your project with technologies such as Apache Airflow or NiFi.

Tomasz Dudek
May 27, 2018 · 9 min read

Software Development vs. AI Project

Traditional Software Development

- **Goal:** Satisfy functional and non-functional requirements (meet a specification)
- **Quality** of the software depends mostly on code
- One software stack (per module) with a limited number of frameworks and tools excluding DevOps
- Usually a single (or few) deployment environments

AI / ML Project

- **Goal:** Optimize a certain metric and try to improve it via experimentation
- **Quality** depends on multiple factors (input data, data cleaning, feature engineering, tuning parameters, ML algorithm, etc.)
- Software stack involves many libraries and frameworks (unclear which is the best)
- Usually diverse deployment environments

AI-enabled software system requires both!

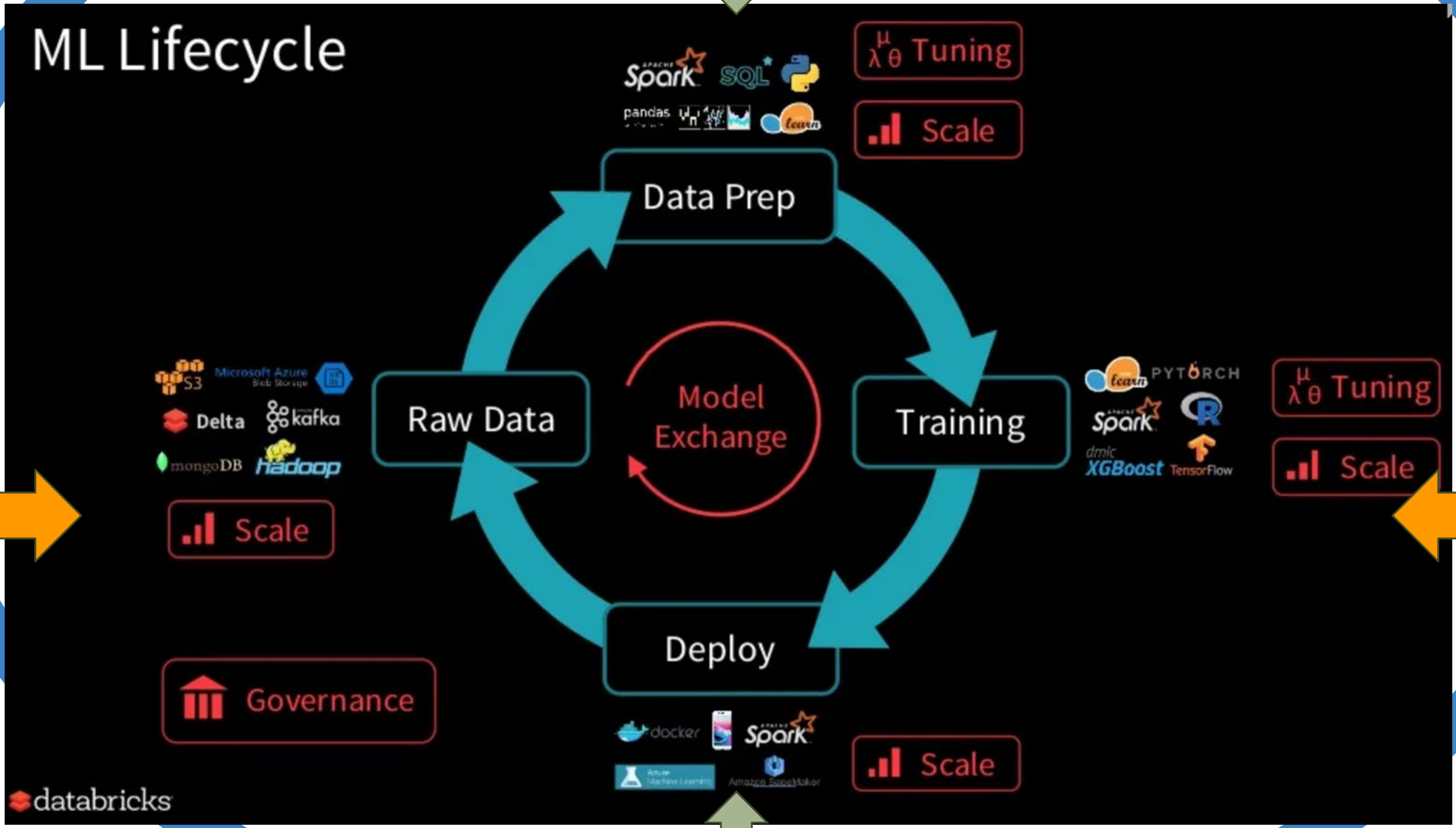
ML Lifecycle

Design + Architecture

MLOps, infrastructure, integrated design

Requirements Engineering

Project requirements, goals, metrics



ML Lifecycle

Data Prep

Training

Deploy

Raw Data

Model Exchange

Tuning

Scale

Tuning

Scale

Scale

Scale

Governance

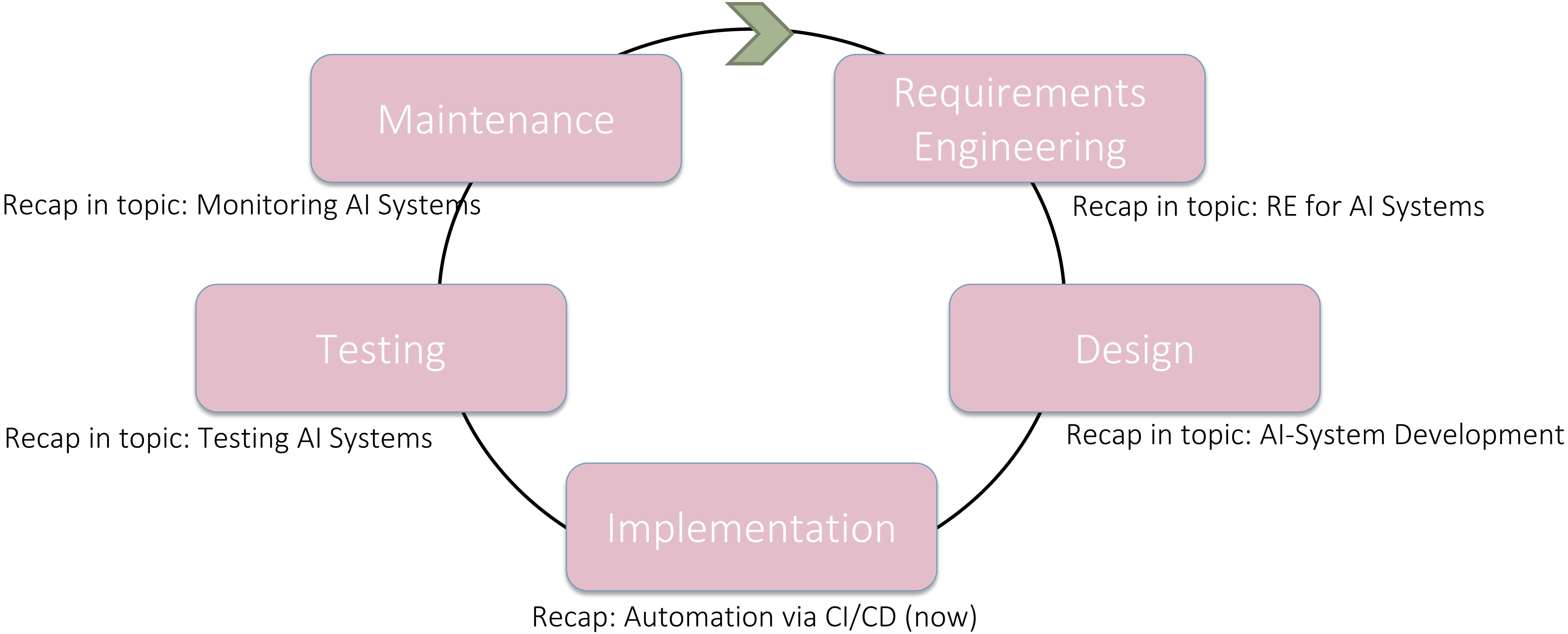
Implementation + Testing

API design, non-functional properties (SLA testing)

Deployment + Monitoring

Deployment infrastructure, heart beat, monitoring/logging capabilities, version management, feedback loop

Iterative development



Recap in topic: Monitoring AI Systems

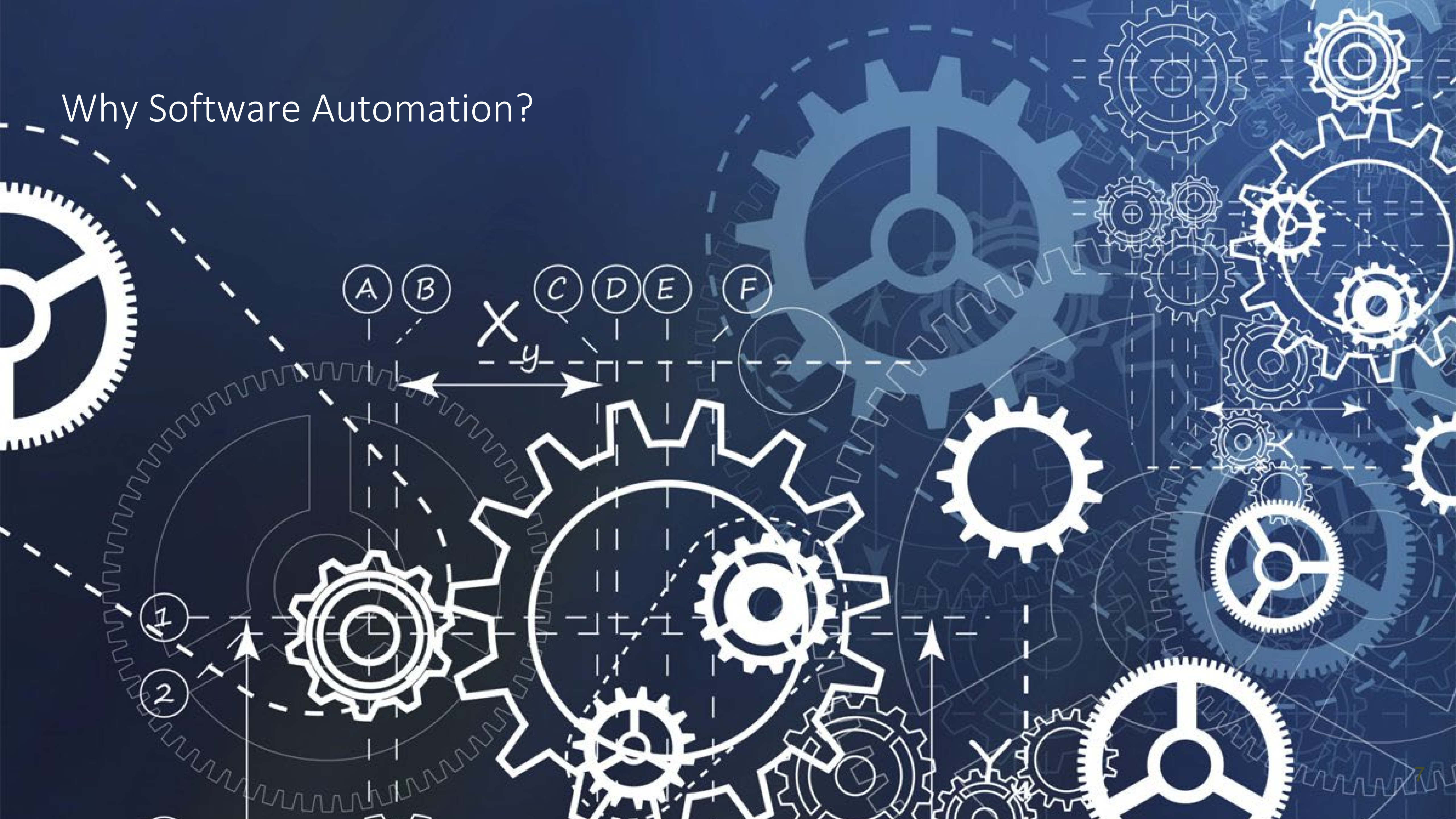
Recap in topic: RE for AI Systems

Recap in topic: Testing AI Systems

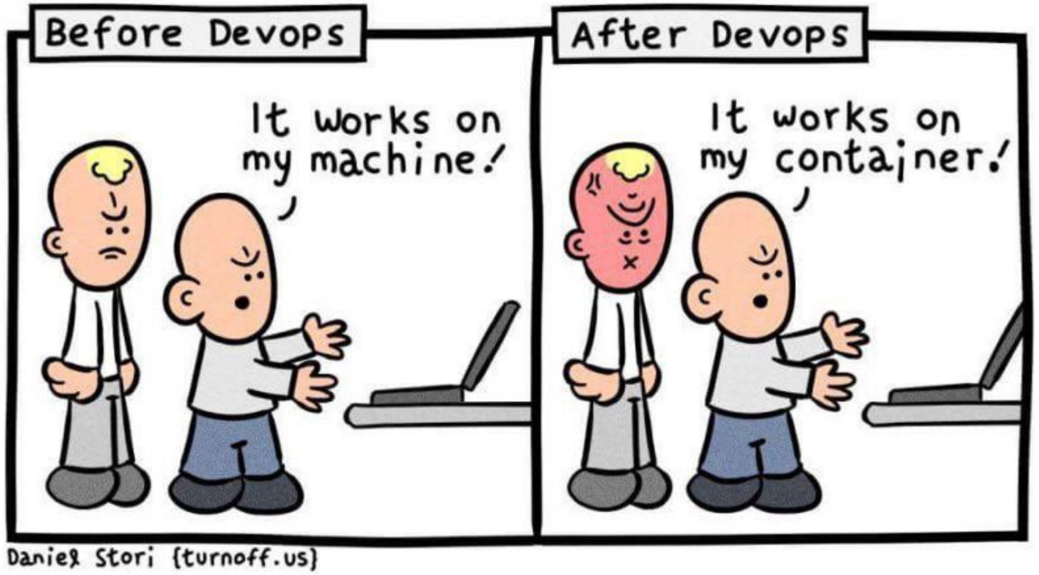
Recap in topic: AI-System Development

Recap: Automation via CI/CD (now)

Why Software Automation?

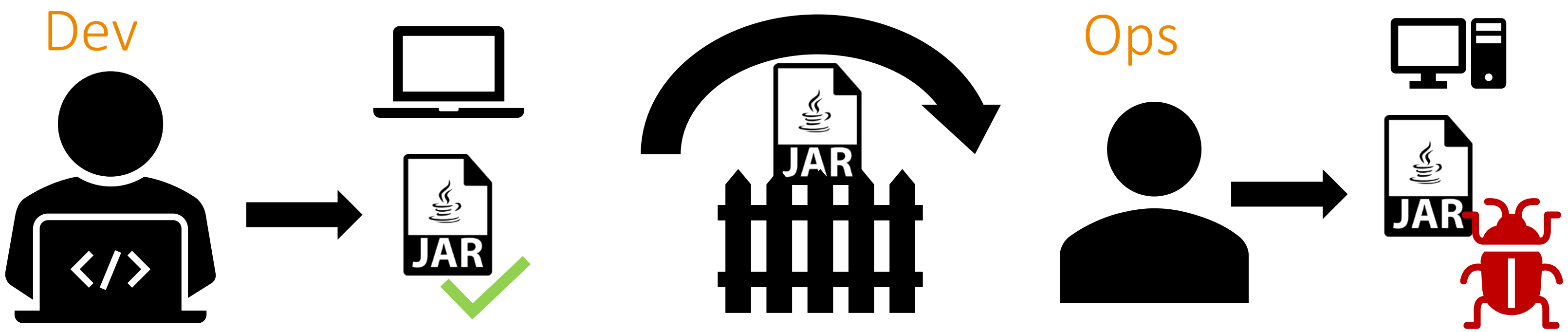


In the old days…



Software systems are developed not in a vacuum:

- Build systems
- JVM versions
- Operating systems (with versions)
- Libraries in the system (with a specific version)
- Data and other sources
- Environment variables
- Hardware specifics (certain processor types or uncommon resources)


Dev

Ops

All this is missing or may be different

What to do?

Developer could test for all possible environments…

Developer could “standardize” the environment…

Developer could recreate her environment at user side…

Developers could ship the steps on how to create the environment in which the software works and encapsulate everything in a sealed world…

Does not scale / infeasible to know

How to enforce? How to update?

Sounds ideal, but how to enforce OS and how to automate correct environment creation?

That’s it!

Two essential developments were necessary: virtualization and pipelines

DevOps Goals

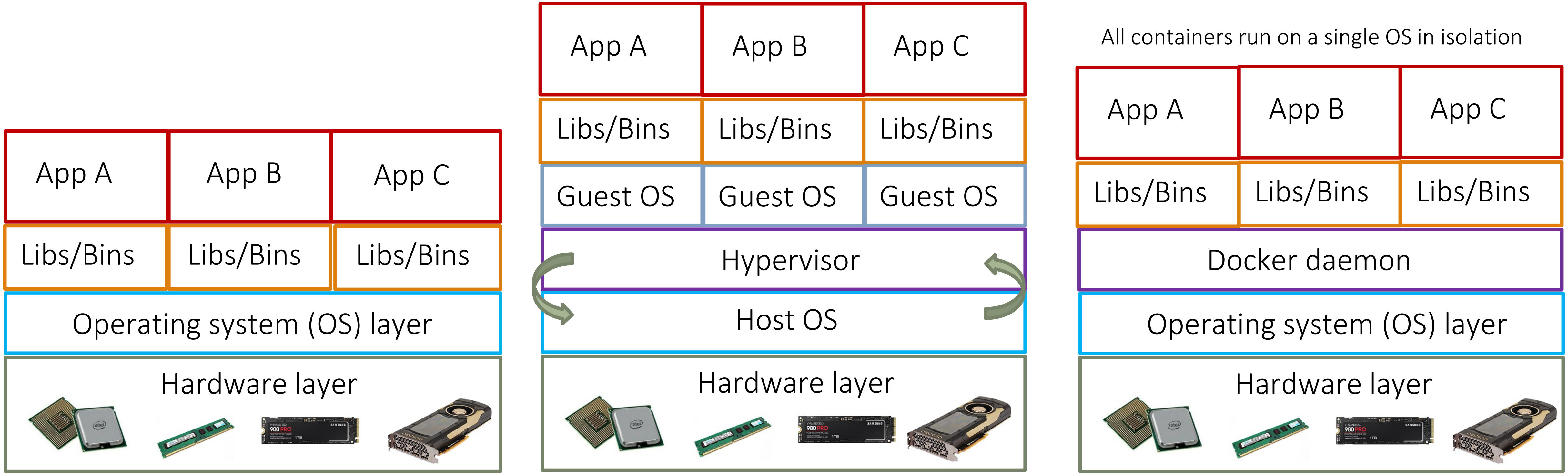
Automate processes (testing, integration, feedback, error report, deployment, etc.) thereby improving software quality through better (more frequent) tests, similar environments in dev and prod, actually needed features (fast feedback), more secure software (using standardized procedures)

Moreover:

- Minimize time to deployment
- Maintain your own product in production
- Enable agile development (iterative process without DevOps possibly infeasible)
- Own the environment (and infrastructure) your system is running on
- Bridge the gap between local and cloud (remote) computing
- Save (reliable) updates and rollouts of software versions

Virtualization and Containerization (simplified)

Idea: Abstract from the (i) physical hardware and (ii) the environment your application is running on. Create your virtual environment anywhere.



App A | App B | App C

Libs/Bins | Libs/Bins | Libs/Bins

Operating system (OS) layer

Hardware layer

Traditional architecture

Have a complete OS running on top of a host OS

App A | App B | App C

Libs/Bins | Libs/Bins | Libs/Bins

Guest OS | Guest OS | Guest OS

Hypervisor

Host OS

Hardware layer

Virtual machines

All containers run on a single OS in isolation

App A | App B | App C

Libs/Bins | Libs/Bins | Libs/Bins

Docker daemon

Operating system (OS) layer

Hardware layer

Containerization

Virtualization at Different Levels

Instruction set architecture	Emulate guest ISA	Dynamo, Bird, Bochs, etc.
Hardware level	Directly on top of HW	VMWare, VirtualPC
OS level	Isolated containers	Docker, Jail, etc.
Run-time library	VM via runtime libs	Wine, vCUDA
Environment	Virtual environment	venv, Conda, etc.
User application	VMs at app level	JVM, .Net

Containers

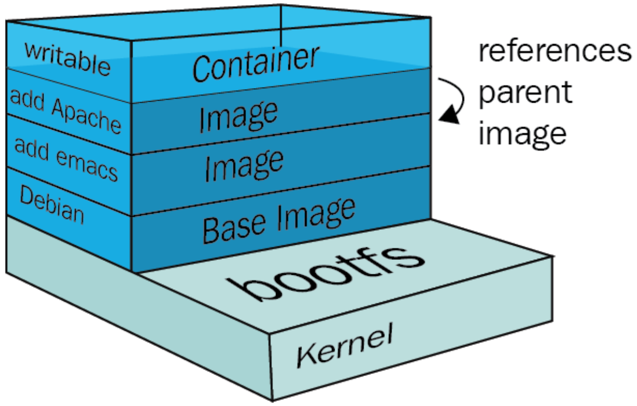
Standardized way of packaging an application with all its dependencies and data such that it can be moved and executed everywhere

Docker creates images for Linux

- Isolated user space within a running Linux kernel
- Shared kernel across containers
- Direct device access
- Isolated runtime and filesystem
- Resource isolation using namespaces

Key feature: Layered images

- Layered filesystem
- Shares common files across layers



Docker Workflow

Dockerfile

A script-like configuration file to specify a base image, the dependencies, environmental parameters, data, and app to run in the container

Docker CLI

A command-line interface to build Docker images from Docker files (binary, shippable representations), list, run, and remove containers.

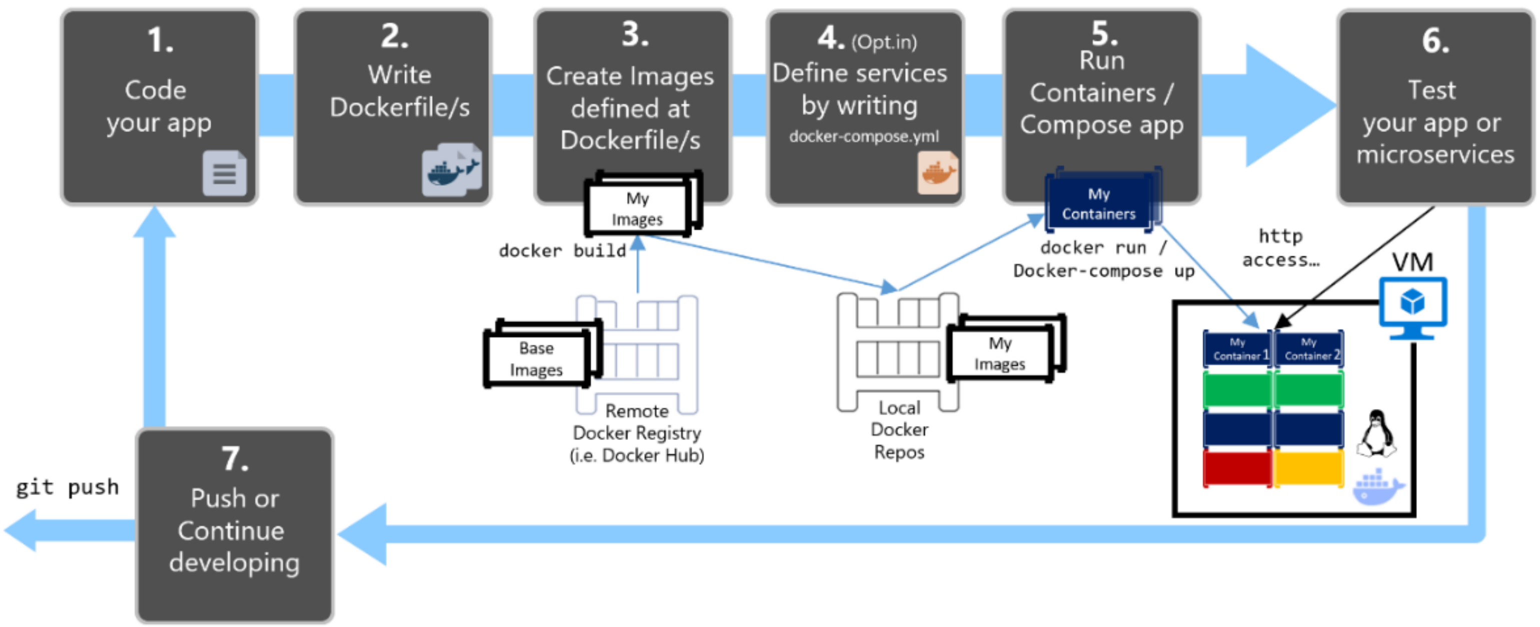
Docker image

A read-only snapshot of a container that is stored in the Docker registry; acts as a template for building containers.

Docker container

The unit in which the app runs.

Inner-Loop development workflow for Docker apps



1. Code your app
2. Write Dockerfile/s
3. Create Images defined at Dockerfile/s
4. (Opt.in) Define services by writing docker-compose.yml
5. Run Containers / Compose app
6. Test your app or microservices
7. Push or Continue developing
git push
My Images
docker build
Base Images
Remote Docker Registry (i.e. Docker Hub)
Local Docker Repos
My Images
My Containers
docker run / Docker-compose up
http access…
VM
My Container 1
My Container 2

Dockerfile Commands (excerpt)

FROM

Define base image to extend (see https://hub.docker.com/explore/ for available images)

```
FROM <image>[:<tag>] [AS <name>]
```

LABEL

Provide key/value metadata (version, topic, etc.)

```
LABEL <key>=<value>
```

ARG

Define Docker build-time variables

```
ARG <name>[=<default value>]
```

WORKDIR

Set the working directory for all following instruction

```
WORKDIR <path>
```

RUN

Executes a command in a new layer on top of the current image and commits the results.

```
RUN <command> (shell form)
RUN ["executable", "param1", "param2"] (exec form)
```

ADD

Copy data from host to container (alternative to COPY)
<dest> is an absolute path or a path relative to WORKDIR

```
ADD [--chown=<user>:<group>] <src>... <dest>
```

ENTRYPOINT

Specifies the executable at container startup

```
ENTRYPOINT ["executable", "param1", "param2"]
```

CMD

Alternative command to run a given executable.

```
CMD ["executable","param1","param2"]
```

Docker CLI Commands

docker build -t <image_tag>:[<version>] docker build -t demo_01 .	Builds the image using the Dockerfile of the current directory (".")
docker image ls or docker images	Lists all the available images
docker run [-d] [--name <name>] [--rm] [-p <ip>:<port_out>:<port_container>] <image_tag> docker run --name tutorial1 --rm demo_01	Create a new instance of the image (run a docker container) [-d] = detached mode (running in background) (use docker attach to revert); [--rm] = remove the container after exits
docker container ls or docker ps	Lists all currently running containers (use [-a] to show stopped as well)
docker image rm <image> docker image rm demo_01	Removes a docker image

Infrastructure as Code (IaC) I

Goal: Automate the configuration, deployment, and management of infrastructure resources, such as virtual machines, containers, networks, and storage

Idea: Define and manage these resources using code. Scripts and templates create, configure, and manage resources.

Benefits:

- Version and document infrastructure in a code repository
- Consistent deployment no matter what the environment or deployment target
- No manual tasks involved (very efficient)
- Scales to hundreds and thousands of resources
- Automates the tasks (e.g., deployment based on events, such as new commits)
- Reproducibility of environments
- Easy sharing or proven templates via tools, such as Terraform, CloudFormation, Ansible

Infrastructure as Code (IaC) II

The whole infrastructure of a project is specified in configuration files

application.properties

bootstrap.yml

pom.xml

Maven

```
[...]

  <artifactId>discovery-microservice</artifactId>
  <version>0.1.0</version>
  <packaging>jar</packaging>

[...]
```

Dockerfile

```
FROM java:8
ADD discovery-microservice-0.1.0.jar app.jar
EXPOSE 8761
ENTRYPOINT ["java","-jar","/app.jar"]
```

Infrastructure as Code (IaC) III

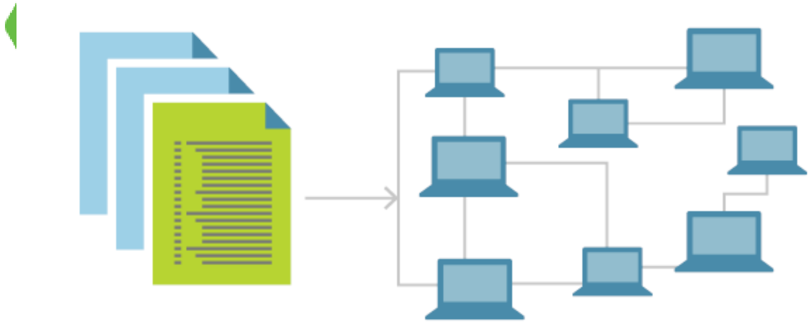
Infrastructure is way more: maintain servers, reconfigure, provision, deploy

application.properties

pom.xml

Maven

```
[...]
  <artifactId>discovery-microservice</artifactId>
  <version>0.1.0</version>
  <packaging>jar</packaging>
[...]
```



ockerfile

```
FROM java:8
ADD discovery-microservice-0.1.0.jar  app.jar
EXPOSE 8761
ENTRYPOINT ["java","-jar","/app.jar"]
```

Goal: Write instructions (code) how the server running your (containerized) app communicate and interact! Deploy your containers automatically and maintain the system in a healthy state (all automated)!

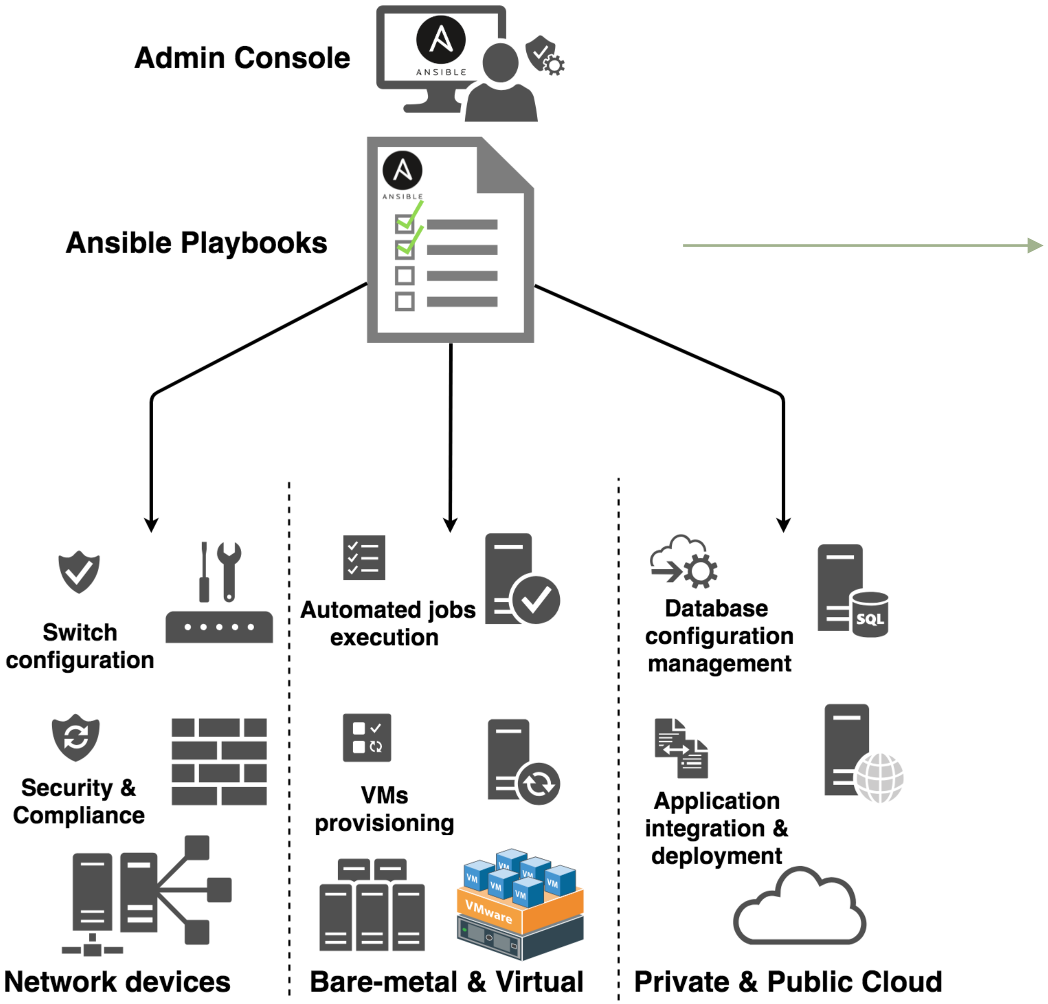
Specifies the underlying infrastructure:

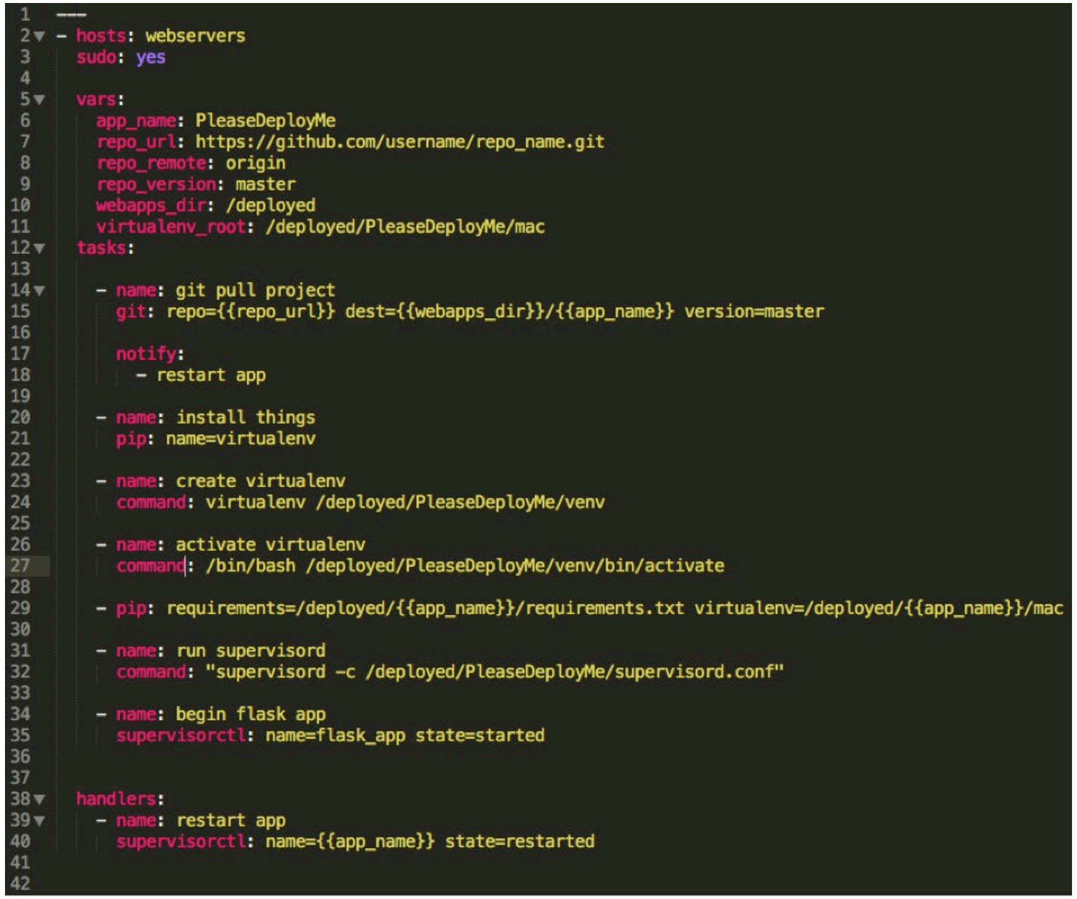
- Software of servers
- Konfiguration of server and software
- Communication among services and server

Key feature:

- Infrastructure is now explicitly specified
- Can be reviewed
- Can be versioned

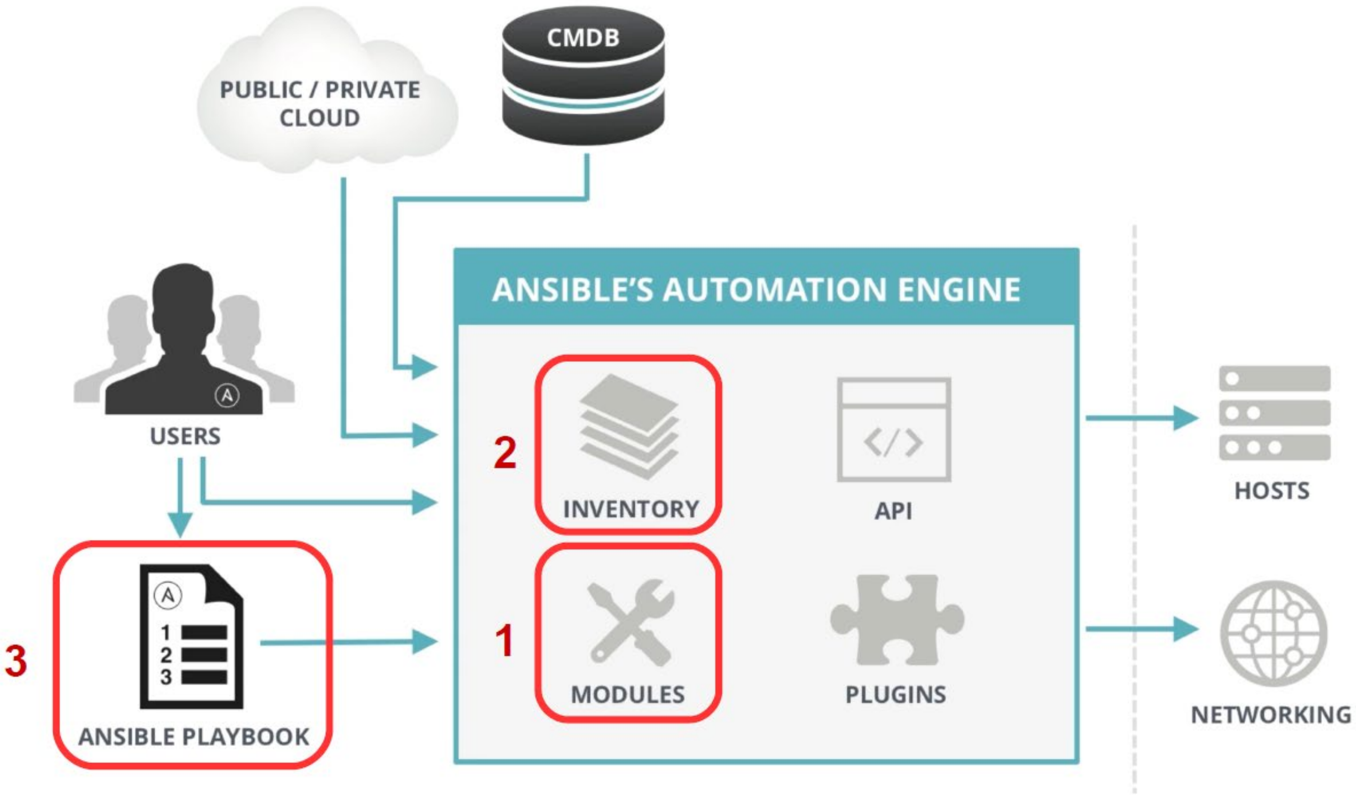
Example: Ansible



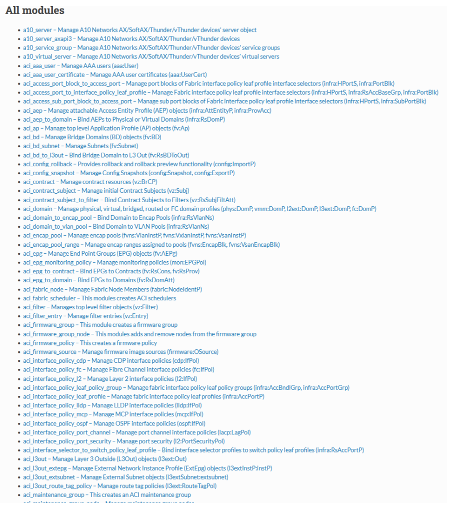


```
---
- hosts: webservers
  sudo: yes

  vars:
    app_name: PleaseDeployMe
    repo_url: https://github.com/username/repo_name.git
    repo_remote: origin
    repo_version: master
    webapps_dir: /deployed
    virtualenv_root: /deployed/PleaseDeployMe/mac
  tasks:

    - name: git pull project
      git: repo={{repo_url}} dest={{webapps_dir}}/{{app_name}} version=master

      notify:
        - restart app

    - name: install things
      pip: name=virtualenv

    - name: create virtualenv
      command: virtualenv /deployed/PleaseDeployMe/venv

    - name: activate virtualenv
      command: /bin/bash /deployed/PleaseDeployMe/venv/bin/activate

    - pip: requirements=/deployed/{{app_name}}/requirements.txt virtualenv=/deployed/{{app_name}}/mac

    - name: run supervisord
      command: "supervisord -c /deployed/PleaseDeployMe/supervisord.conf"

    - name: begin flask app
      supervisorctl: name=flask_app state=started


  handlers:
    - name: restart app
      supervisorctl: name={{app_name}} state=restarted


```

Ansible Architecture



(1) Ansible modules: Small scripts pushed from server (control machine) to clients; executed at clients (Hosts) Modules control system resources, such as services, packages, or files and handle executing system commands.



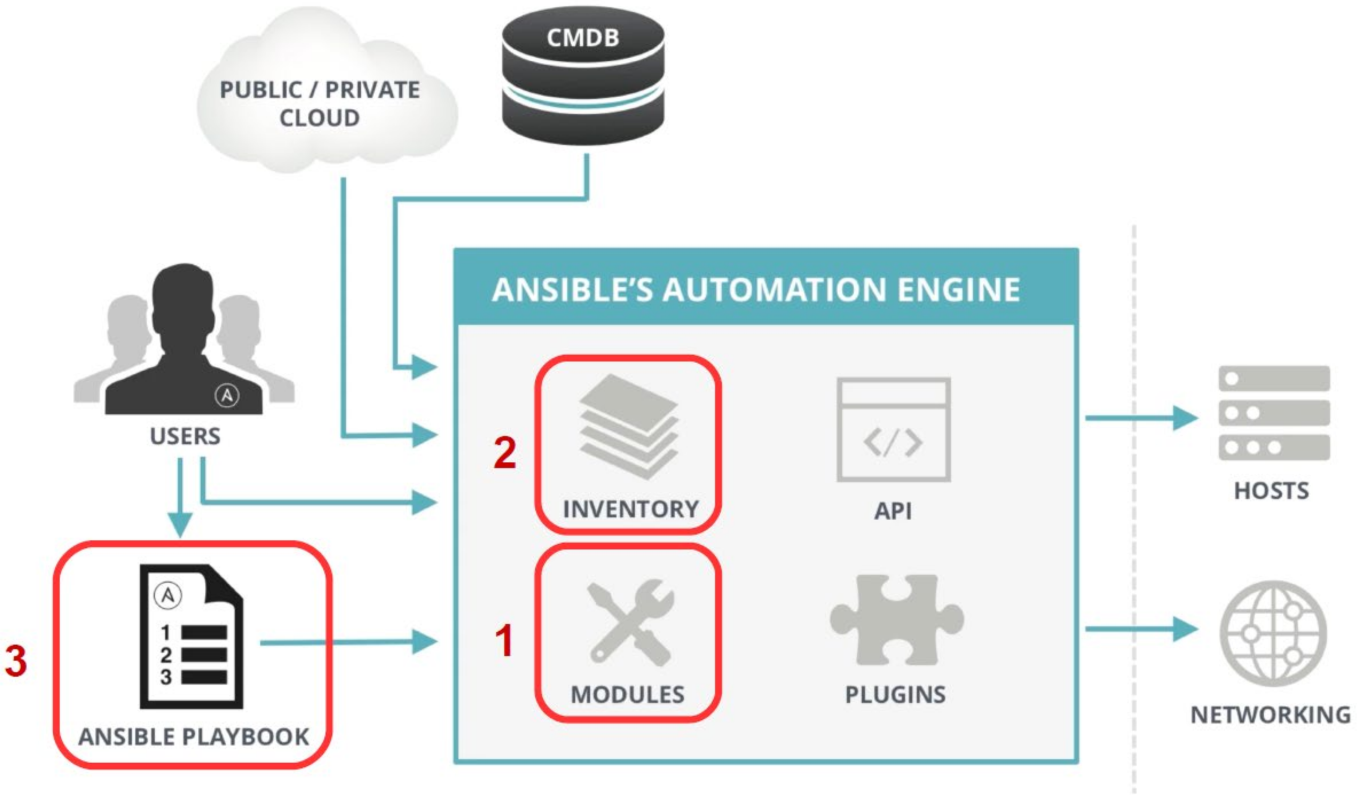
All modules

- a10_server – Manage A10 Networks AX/SoftAX/Thunder/vThunder devices' server object
- a10_server_axapi3 – Manage A10 Networks AX/SoftAX/Thunder/vThunder devices
- a10_service_group – Manage A10 Networks AX/SoftAX/Thunder/vThunder devices' service groups
- a10_virtual_server – Manage A10 Networks AX/SoftAX/Thunder/vThunder devices' virtual servers
- aci_aaa_user – Manage AAA users (aaa:User)
- aci_aaa_user_certificate – Manage AAA user certificates (aaa:UserCert)
- aci_access_port_block_to_access_port – Manage port blocks of Fabric interface policy leaf profile interface selectors (infra:HPortS, infra:PortBlk)
- aci_access_port_to_interface_policy_leaf_profile – Manage Fabric interface policy leaf profile interface selectors (infra:HPortS, infra:RsAccBaseGrp, infra:PortBlk)
- aci_access_sub_port_block_to_access_port – Manage sub port blocks of Fabric interface policy leaf profile interface selectors (infra:HPortS, infra:SubPortBlk)
- aci_aep – Manage attachable Access Entity Profile (AEP) objects (infra:AttEntityP, infra:ProvAcc)
- aci_aep_to_domain – Bind AEPs to Physical or Virtual Domains (infra:RsDomP)
- aci_ap – Manage top level Application Profile (AP) objects (fv:Ap)
- aci_bd – Manage Bridge Domains (BD) objects (fv:BD)
- aci_bd_subnet – Manage Subnets (fv:Subnet)
- aci_bd_to_l3out – Bind Bridge Domain to L3 Out (fv:RsBDToOut)
- aci_config_rollback – Provides rollback and rollback preview functionality (config:ImportP)
- aci_config_snapshot – Manage Config Snapshots (config:Snapshot, config:ExportP)
- aci_contract – Manage contract resources (vz:BrCP)
- aci_contract_subject – Manage initial Contract Subjects (vz:Subj)
- aci_contract_subject_to_filter – Bind Contract Subjects to Filters (vz:RsSubjFiltAtt)
- aci_domain – Manage physical, virtual, bridged, routed or FC domain profiles (phys:DomP, vmm:DomP, l2ext:DomP, l3ext:DomP, fc:DomP)
- aci_domain_to_encap_pool – Bind Domain to Encap Pools (infra:RsVlanNs)
- aci_domain_to_vlan_pool – Bind Domain to VLAN Pools (infra:RsVlanNs)
- aci_encap_pool – Manage encap pools (fvns:VlanInstP, fvns:VxlanInstP, fvns:VsanInstP)
- aci_encap_pool_range – Manage encap ranges assigned to pools (fvns:EncapBlk, fvns:VsanEncapBlk)
- aci_epg – Manage End Point Groups (EPG) objects (fv:AEPg)
- aci_epg_monitoring_policy – Manage monitoring policies (mon:EPGPol)
- aci_epg_to_contract – Bind EPGs to Contracts (fv:RsCons, fv:RsProv)
- aci_epg_to_domain – Bind EPGs to Domains (fv:RsDomAtt)
- aci_fabric_node – Manage Fabric Node Members (fabric:NodeIdentP)
- aci_fabric_scheduler – This modules creates ACI schedulers
- aci_filter – Manages top level filter objects (vz:Filter)
- aci_filter_entry – Manage filter entries (vz:Entry)
- aci_firmware_group – This module creates a firmware group
- aci_firmware_group_node – This modules adds and remove nodes from the firmware group
- aci_firmware_policy – This creates a firmware policy
- aci_firmware_source – Manage firmware image sources (firmware:OSource)
- aci_interface_policy_cdp – Manage CDP interface policies (cdp:IfPol)
- aci_interface_policy_fc – Manage Fibre Channel interface policies (fc:IfPol)
- aci_interface_policy_l2 – Manage Layer 2 interface policies (l2:IfPol)
- aci_interface_policy_leaf_policy_group – Manage fabric interface policy leaf policy groups (infra:AccBndlGrp, infra:AccPortGrp)
- aci_interface_policy_leaf_profile – Manage fabric interface policy leaf profiles (infra:AccPortP)
- aci_interface_policy_lldp – Manage LLDP interface policies (lldp:IfPol)
- aci_interface_policy_mcp – Manage MCP interface policies (mcp:IfPol)
- aci_interface_policy_ospf – Manage OSPF interface policies (ospf:IfPol)
- aci_interface_policy_port_channel – Manage port channel interface policies (lacp:LagPol)
- aci_interface_policy_port_security – Manage port security (l2:PortSecurityPol)
- aci_interface_selector_to_switch_policy_leaf_profile – Bind interface selector profiles to switch policy leaf profiles (infra:RsAccPortP)
- aci_l3out – Manage Layer 3 Outside (L3Out) objects (l3ext:Out)
- aci_l3out_extepg – Manage External Network Instance Profile (ExtEpg) objects (l3extInstP:instP)
- aci_l3out_extsubnet – Manage External Subnet objects (l3extSubnet:extsubnet)
- aci_l3out_route_tag_policy – Manage route tag policies (l3ext:RouteTagPol)
- aci_maintenance_group – This creates an ACI maintenance group
- aci_maintenance_group_node – Manage maintenance group nodes

https://docs.ansible.com/ansible/2.9/modules/list_of_all_modules.html

Users can write their own modules via Python

Ansible Architecture

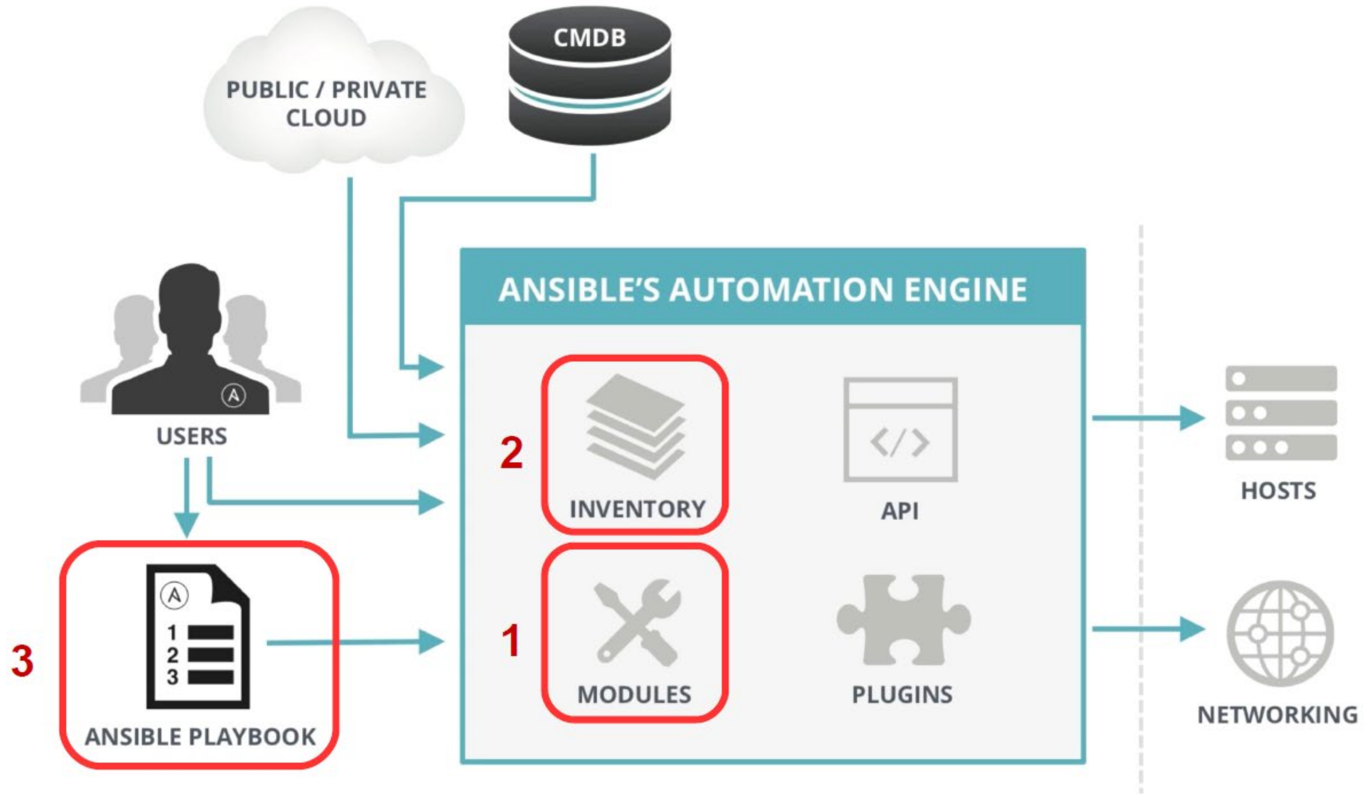


(2) List of Host machines (IP addresses) including grouping capabilities.

```
mail.example.com

[webservers]
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
three.example.com
```

Patterns in the Playbook can target specific hosts or groups with actions (pushed modules).

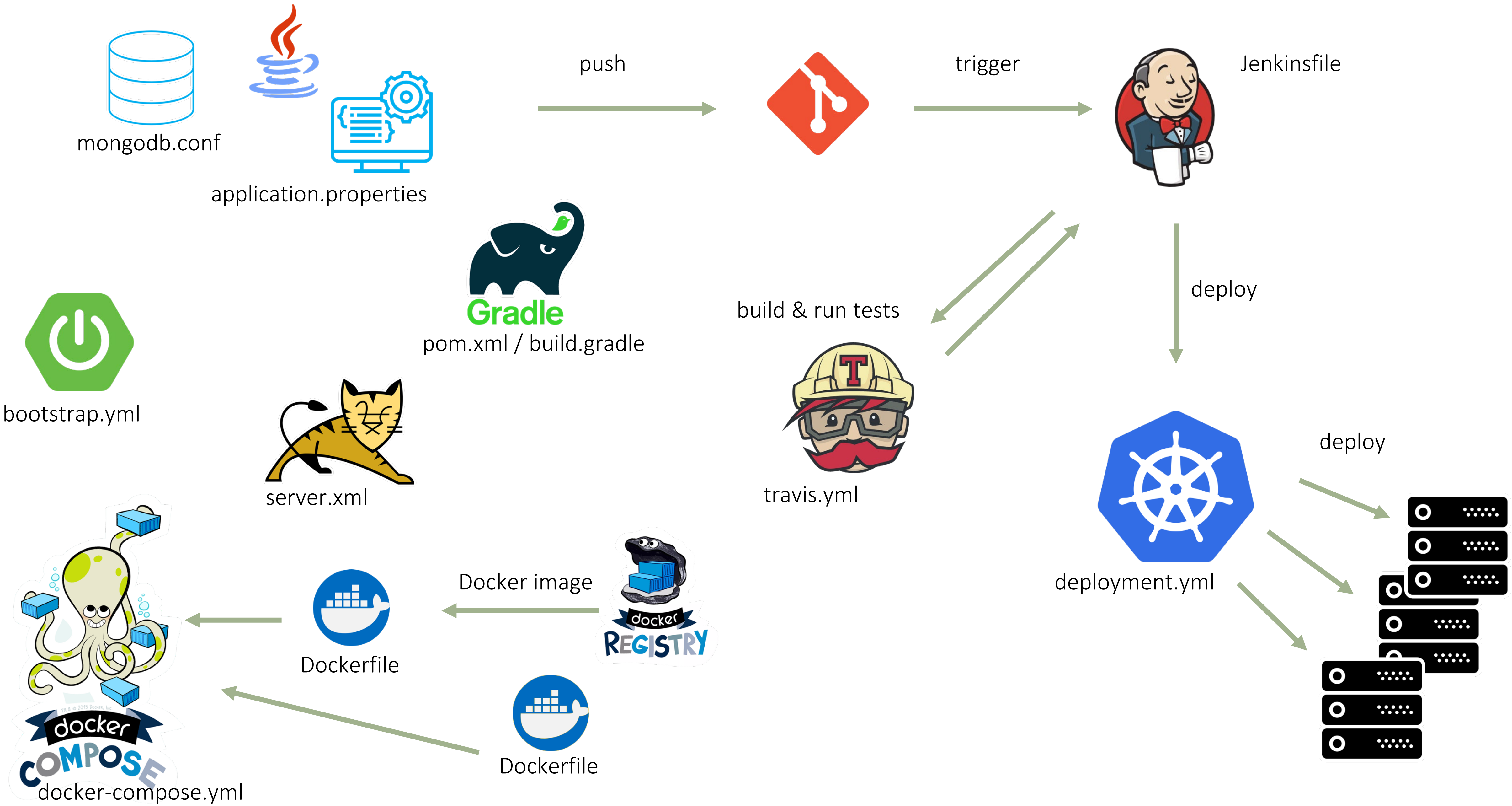
Ansible Architecture



(3) Contain instructions about what to do (tasks) and where (hosts).
Declare configurations, orchestrate steps to provision a server, launch tasks synchronous and async.
Written in YAML and composed of multiple „plays“.
A play maps a (group of) host(s) to some roles called tasks whereas a task refers to a Ansible module.

```
---
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum:
      name: httpd
      state: latest
  - name: write the apache config file
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running
    service:
      name: httpd
      state: started
  handlers:
    - name: restart apache
      service:
        name: httpd
        state: restarted
```

Configuration Hell: Mind the Technical Debt!



mongodb.conf

application.properties

push

trigger

Jenkinsfile

pom.xml / build.gradle

bootstrap.yml

build & run tests

deploy

server.xml

travis.yml

deployment.yml

deploy

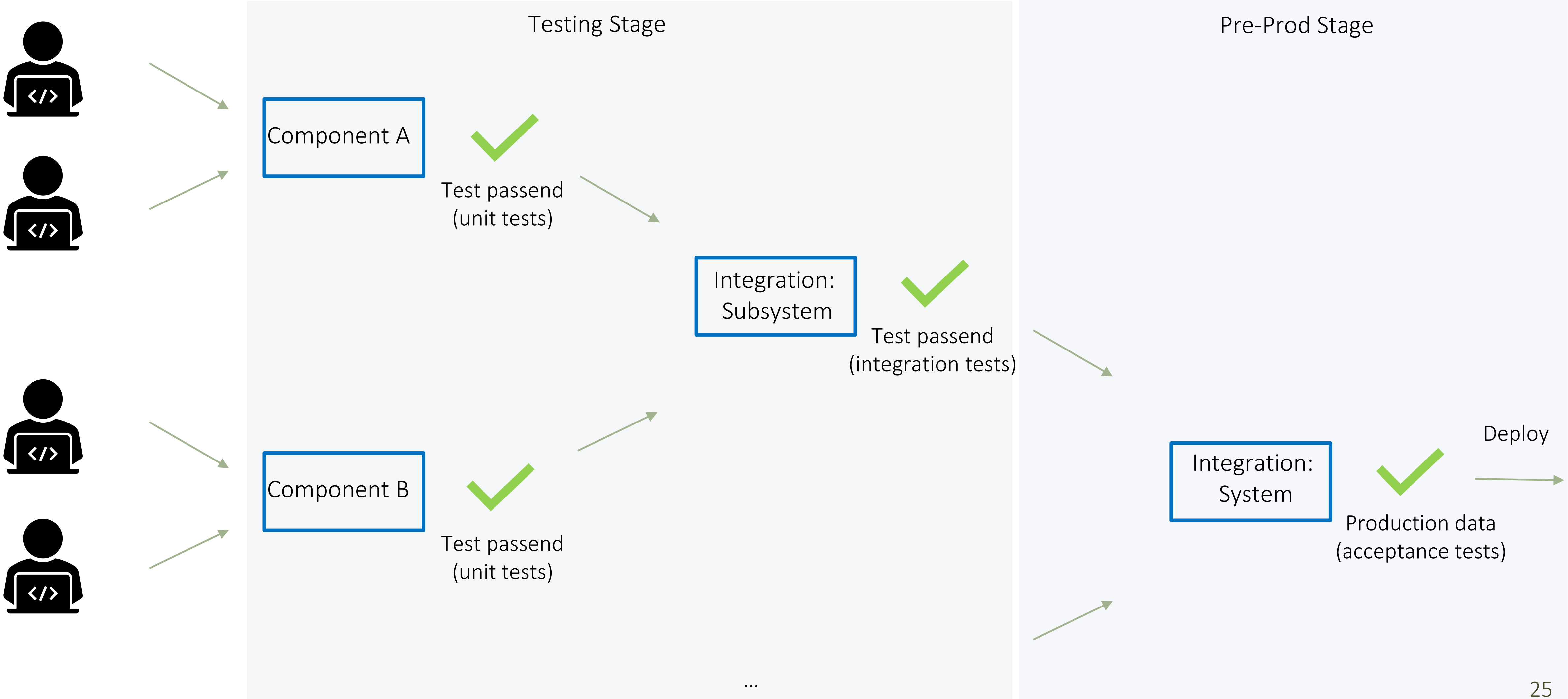
Docker image

Dockerfile

Dockerfile

docker-compose.yml

Continuous Integration and Delivery (CI/CD)



Testing Stage

Pre-Prod Stage

Component A

Test passend
(unit tests)

Component B

Test passend
(unit tests)

Integration:
Subsystem

Test passend
(integration tests)

Integration:
System

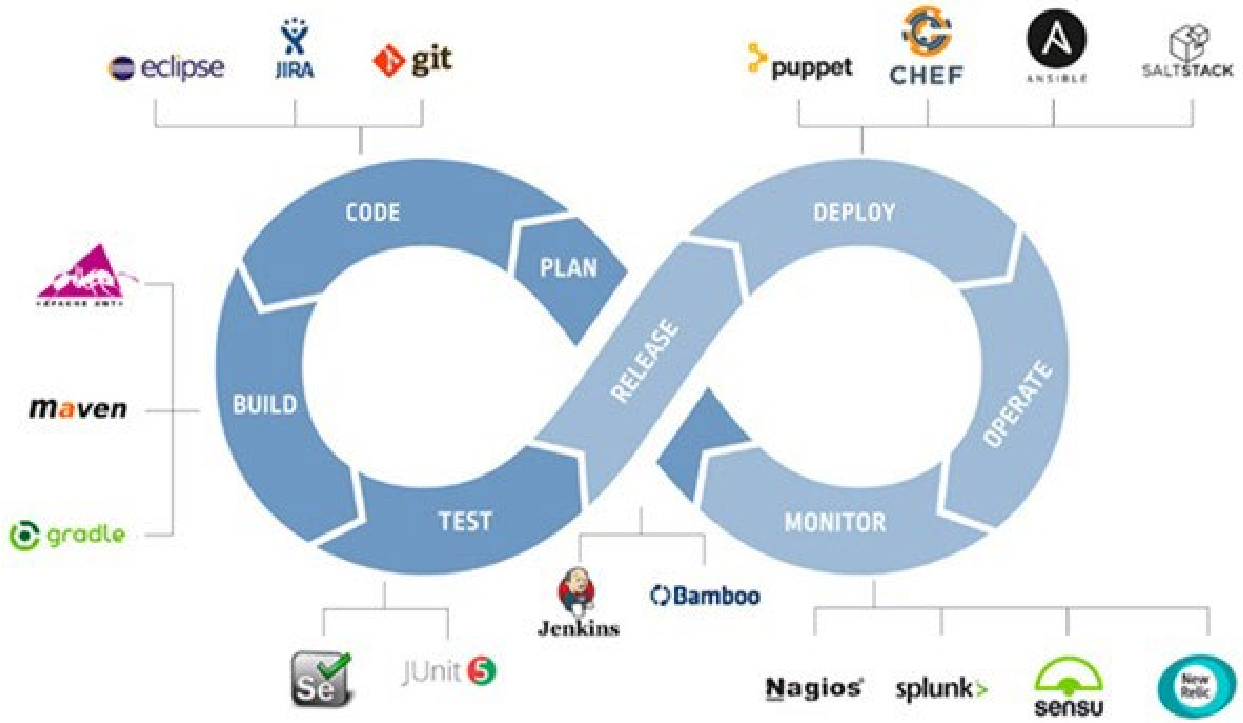
Production data
(acceptance tests)

Deploy

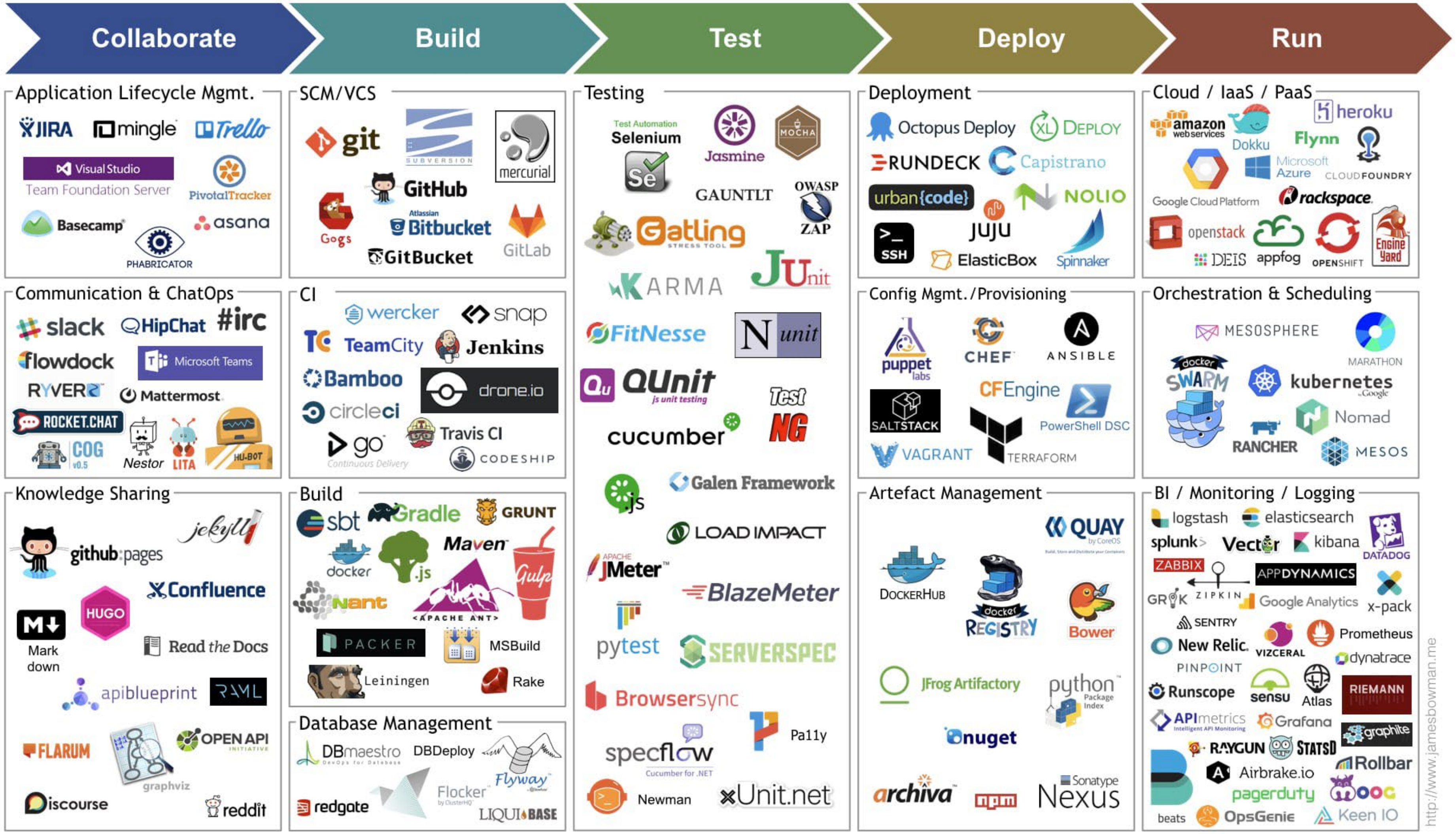
…

CI/CD goal: Automate the whole pipeline such that each push triggers the CI/CD process

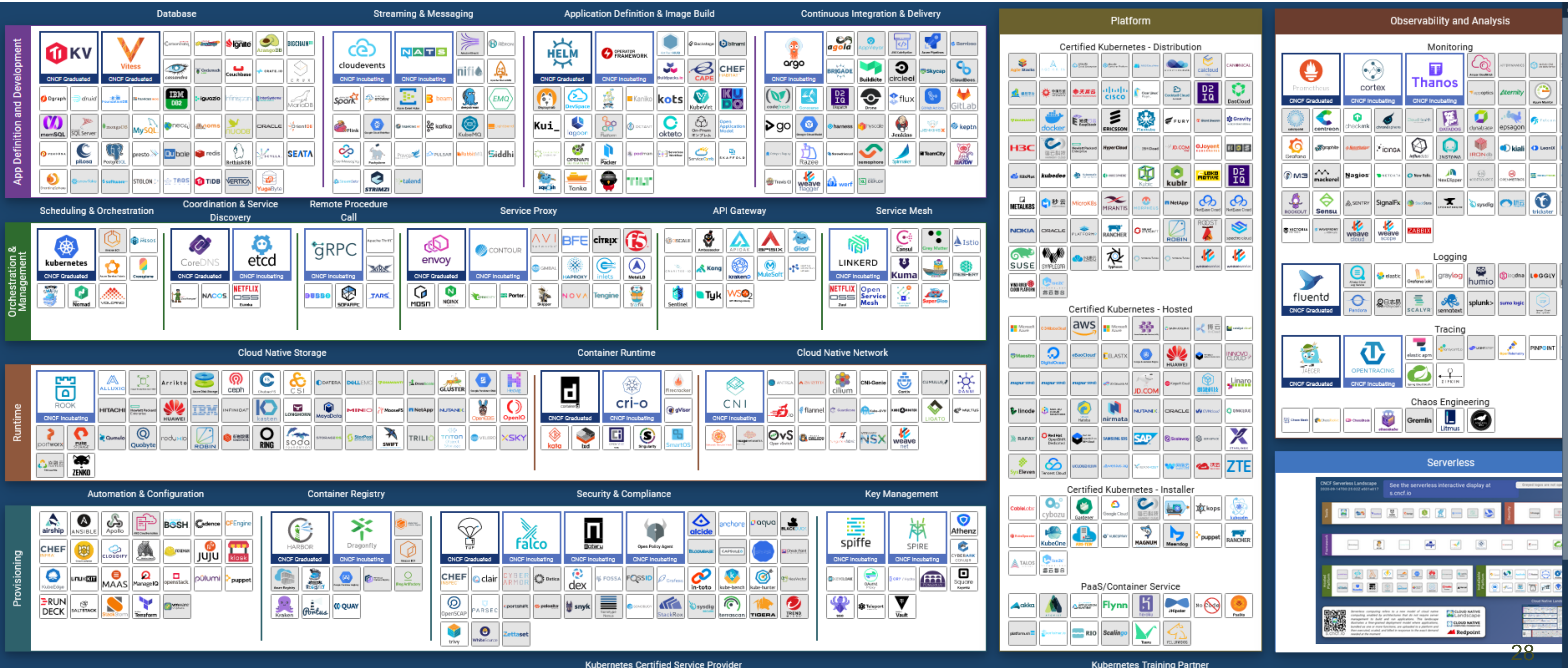
Tools in a CI/CD Pipeline



Small Excerpt of Available Tools



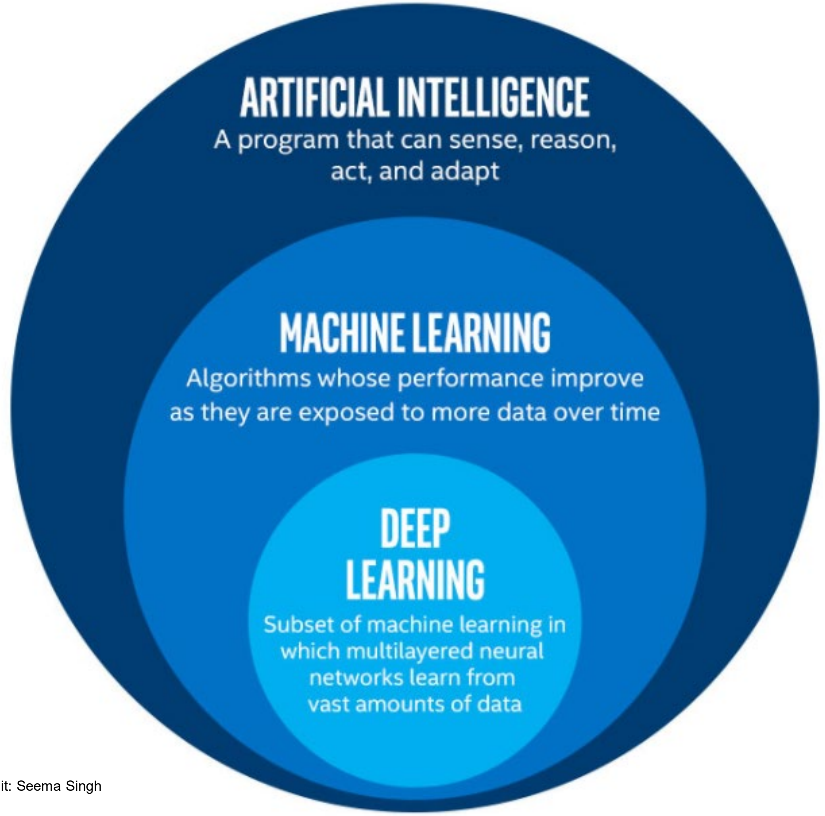
Still not all… but ML Systems make no difference!


App Definition and Development
Database
Streaming & Messaging
Application Definition & Image Build
Continuous Integration & Delivery
Orchestration & Management
Scheduling & Orchestration
Coordination & Service Discovery
Remote Procedure Call
Service Proxy
API Gateway
Service Mesh
Runtime
Cloud Native Storage
Container Runtime
Cloud Native Network
Provisioning
Automation & Configuration
Container Registry
Security & Compliance
Key Management
Platform
Certified Kubernetes - Distribution
Certified Kubernetes - Hosted
Certified Kubernetes - Installer
PaaS/Container Service
Observability and Analysis
Monitoring
Logging
Tracing
Chaos Engineering
Serverless
Kubernetes Certified Service Provider
Kubernetes Training Partner
CNCF Graduated
CNCF Incubating

Topic II:

Basics of ML/AI Development

AI & Machine Learning



ARTIFICIAL INTELLIGENCE

A program that can sense, reason, act, and adapt

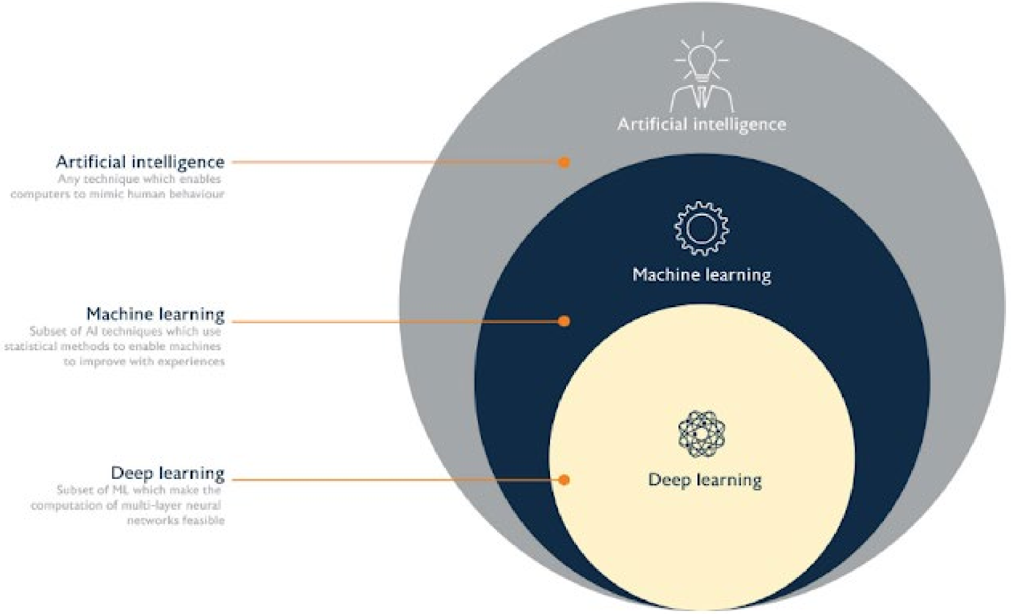
MACHINE LEARNING

Algorithms whose performance improve as they are exposed to more data over time

DEEP LEARNING

Subset of machine learning in which multilayered neural networks learn from vast amounts of data

Credit: Seema Singh



Artificial intelligence

Any technique which enables computers to mimic human behaviour

Machine learning

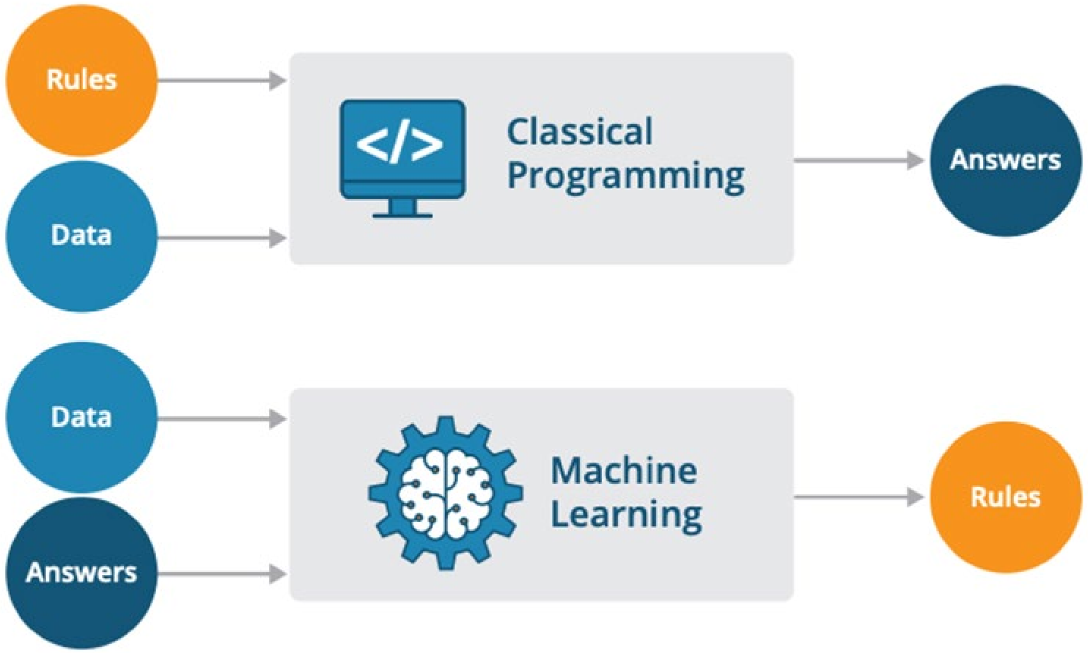
Subset of AI techniques which use statistical methods to enable machines to improve with experiences

Deep learning

Subset of ML which make the computation of multi-layer neural networks feasible

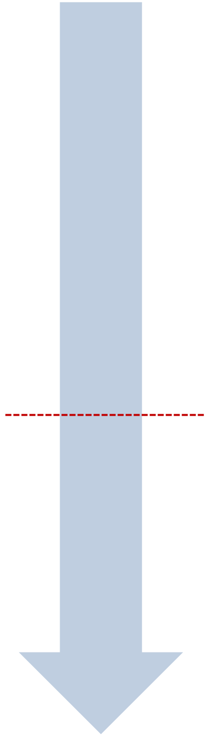
Credit: BlogInsider

Programming versus Machine Learning



Credit: Sravya Reddysetty

What is “Intelligence”?

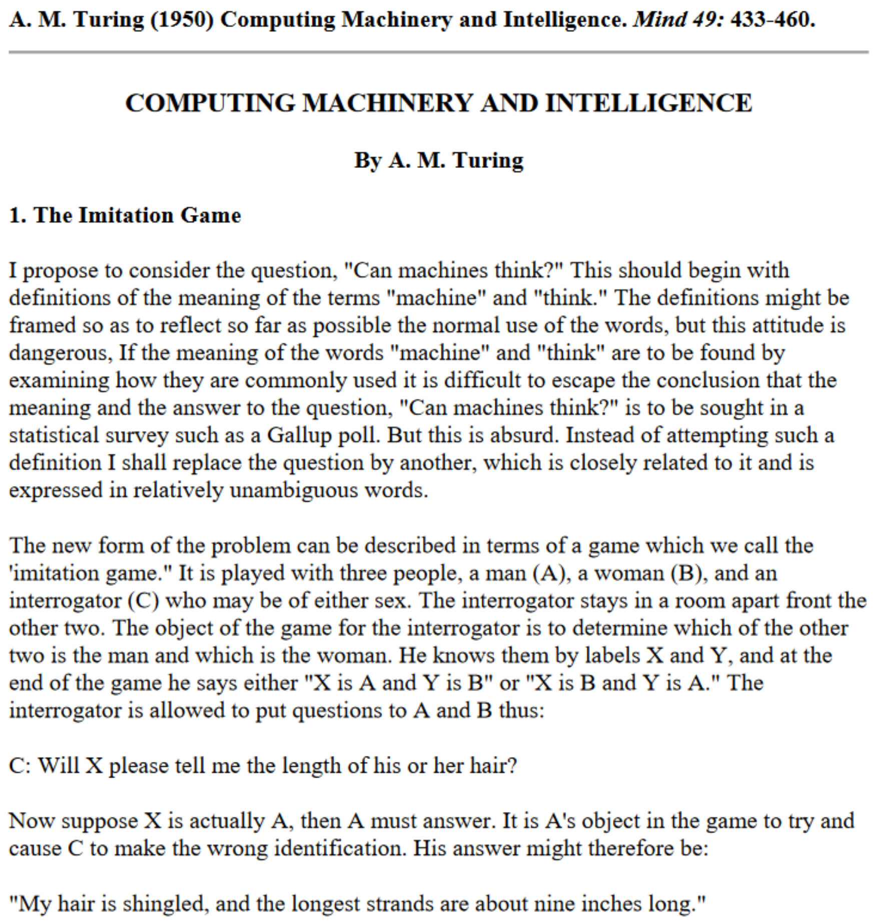


Reactive machines — Designed for a specific task; perceives & reacts to the current world in the same way for the same input

Limited memory — Store previous data and predictions to weigh future decisions (e.g., language models, reinforcement learning)

Theory of mind — Understand that others have thought and emotions. Comprehend how humans would feel

Self-awareness — Consciousness + how to replicate



A. M. Turing (1950) Computing Machinery and Intelligence. *Mind 49:* 433-460.

COMPUTING MACHINERY AND INTELLIGENCE

By A. M. Turing

1. The Imitation Game

I propose to consider the question, "Can machines think?" This should begin with definitions of the meaning of the terms "machine" and "think." The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous, If the meaning of the words "machine" and "think" are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, "Can machines think?" is to be sought in a statistical survey such as a Gallup poll. But this is absurd. Instead of attempting such a definition I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words.

The new form of the problem can be described in terms of a game which we call the 'imitation game." It is played with three people, a man (A), a woman (B), and an interrogator (C) who may be of either sex. The interrogator stays in a room apart front the other two. The object of the game for the interrogator is to determine which of the other two is the man and which is the woman. He knows them by labels X and Y, and at the end of the game he says either "X is A and Y is B" or "X is B and Y is A." The interrogator is allowed to put questions to A and B thus:

C: Will X please tell me the length of his or her hair?

Now suppose X is actually A, then A must answer. It is A's object in the game to try and cause C to make the wrong identification. His answer might therefore be:

"My hair is shingled, and the longest strands are about nine inches long."

Elements of AI/ML Model Usage

Context

Information to be processed →

Information at runtime

- System state (e.g., lights, variables)
- User / profile information
- Intended use case triggered
- Sensors, environmental data
- Explicit data (e.g., camera)
- Interaction items of user (e.g., query)
- Interaction history (e.g., visited sites)

AI / ML model

Model produces result →

Information at design time

- Web scraping
- Public data
- User reports / recommendations
- Historical sensor data
- Collected images
- Google BigQuery

Output

Types of ML

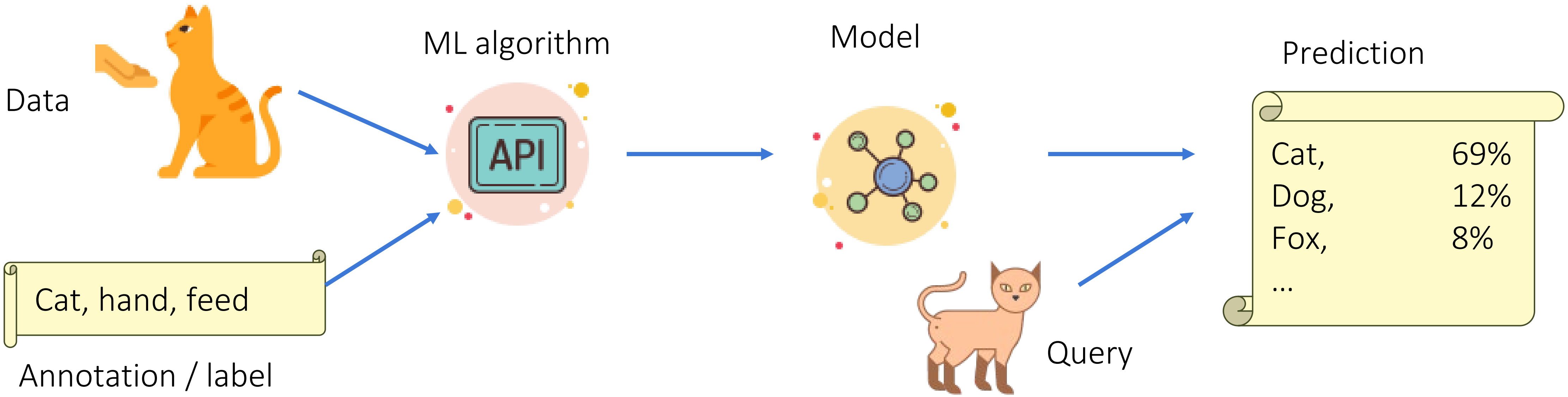
- Clustering
- Classification
- Probability estimates
- Ranking
- Recommendations
- Hybrid (e.g., NLP)

What is important? What does the model need?
What to store? What is legal, unbiased, and ethical?

Keep design time and runtime information at synch!

Types of ML

Supervised Learning



Data

Cat, hand, feed

Annotation / label

ML algorithm

API

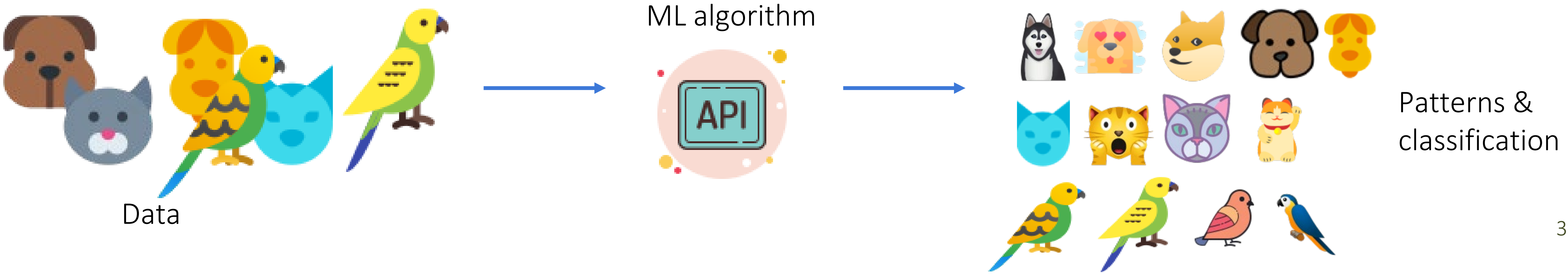
Model

Prediction

Cat,	69%
Dog,	12%
Fox,	8%
…	

Query

Unsupervised Learning



Data

ML algorithm

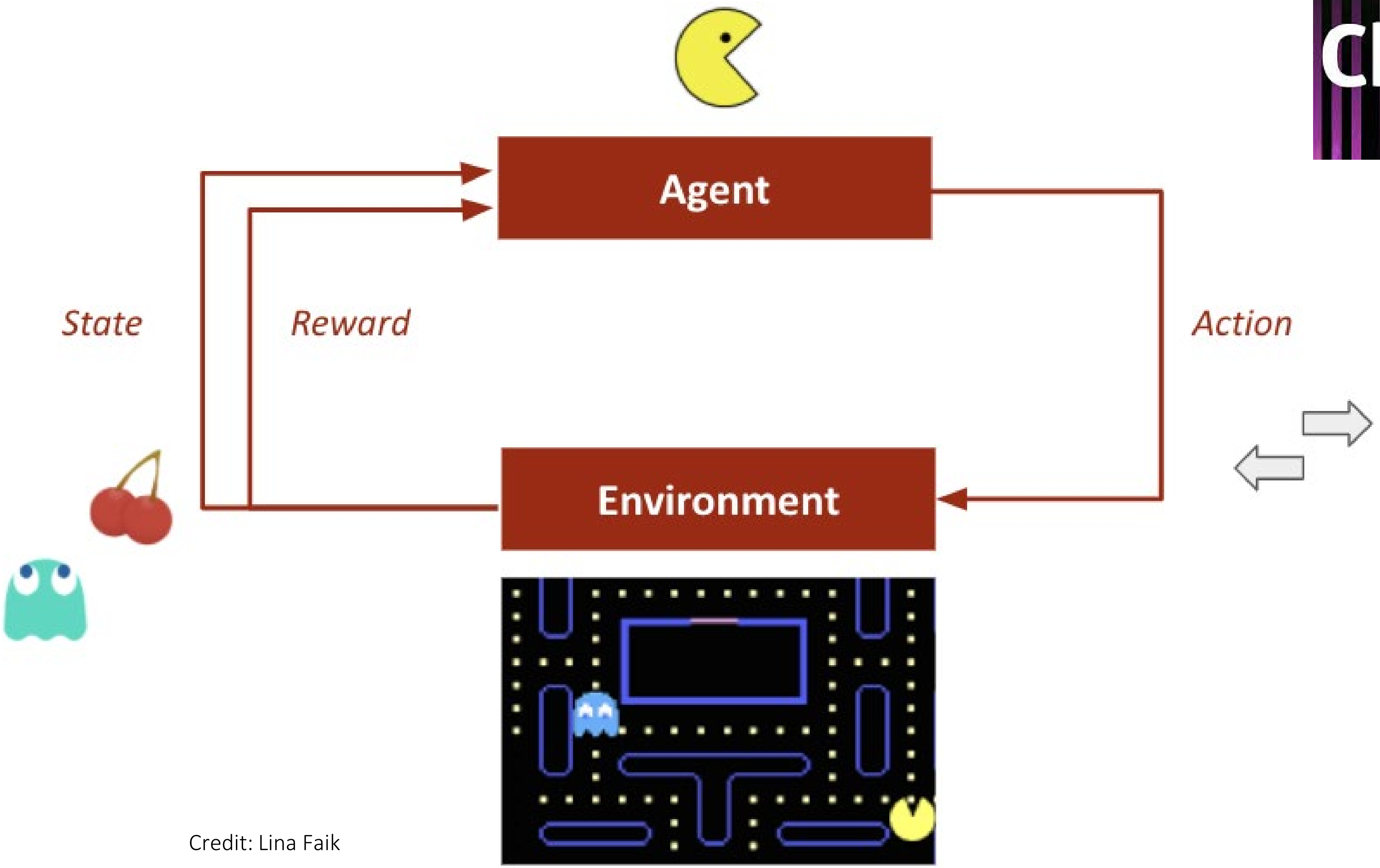
API

Patterns & classification

Types of ML II

Reinforcement Learning

ChatGPT



Credit: Lina Faik

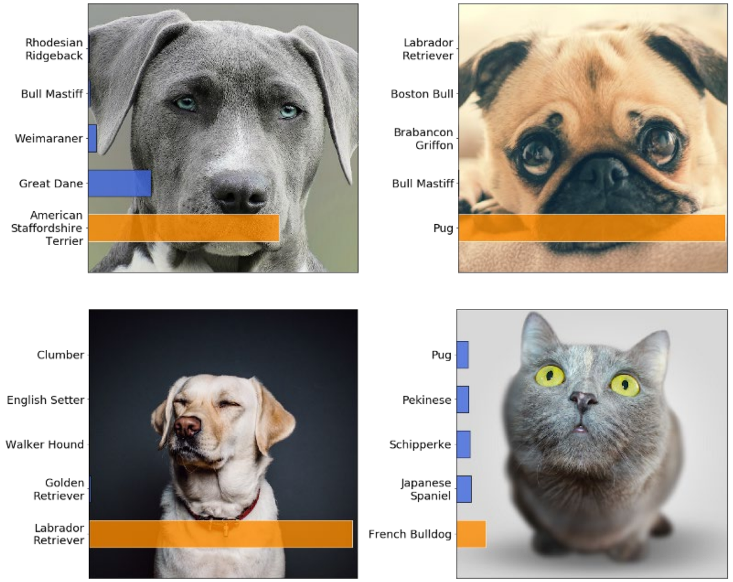
ML/AI Archetypes I

Classification

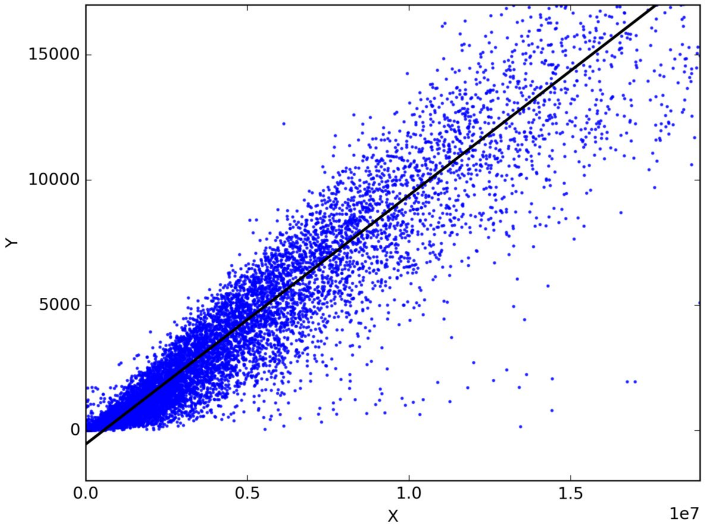
- Classify entities based on the given data (context, input, features)
- Usually applied to few classes (e.g., spam or no spam)

Probability estimates

- Predict probability of a certain outcome or type of input (similar to classification)
- Provides an estimate how certain the model is
- Can work with many outcomes
- Can be transferred to classification when applying a threshold



ML/AI Archetypes II



Regression

- Predict a continuous number (rather than from a finite set)
- Extra- and interpolates among seen data points (generalizes)
- Difficult to keep the model stable and balance bias-variance trade-off (see later)

Rankings

- Ranks the results of the query according to a defined metric (e.g., likelihood of following a link)
- Search results, product rankings, music and film ordering at web page
- Ranking algorithms are more sophisticated than just probabilities

thispersondoesnotexist.com

ML/AI Archetypes III

Generative models

- Generate non-trivial data often based on input information
- Wide range of outputs: language models, style transfer, image generation
- Less constrained in output increases risks in production (not first choice)

Catalog organization

- Models produce a set of results, possibly conditioned on an input
- Similar to a collaborative and content-based recommendation system, but can have capabilities for search a catalog