Continuous Experimentation

Release Engineering for Machine Learning Applications

(REMLA, CS4295)

Sebastian Proksch
S.Proksch@tudelft.nl

Luís Cruz
L.Cruz@tudelft.nl

Goal of today...

- Rationale behind continuous experimentation
- Technical realization of continuous experimentation
- Different implementation strategies
- Role of monitoring
- Evolution of continuous experimentation

What is the job of a Software Engineer?

What is the job of a Software Engineer?

~~Pump out source code~~

~~Implement features from backlog~~

~~Well-designed architecture~~

~~Properly test~~

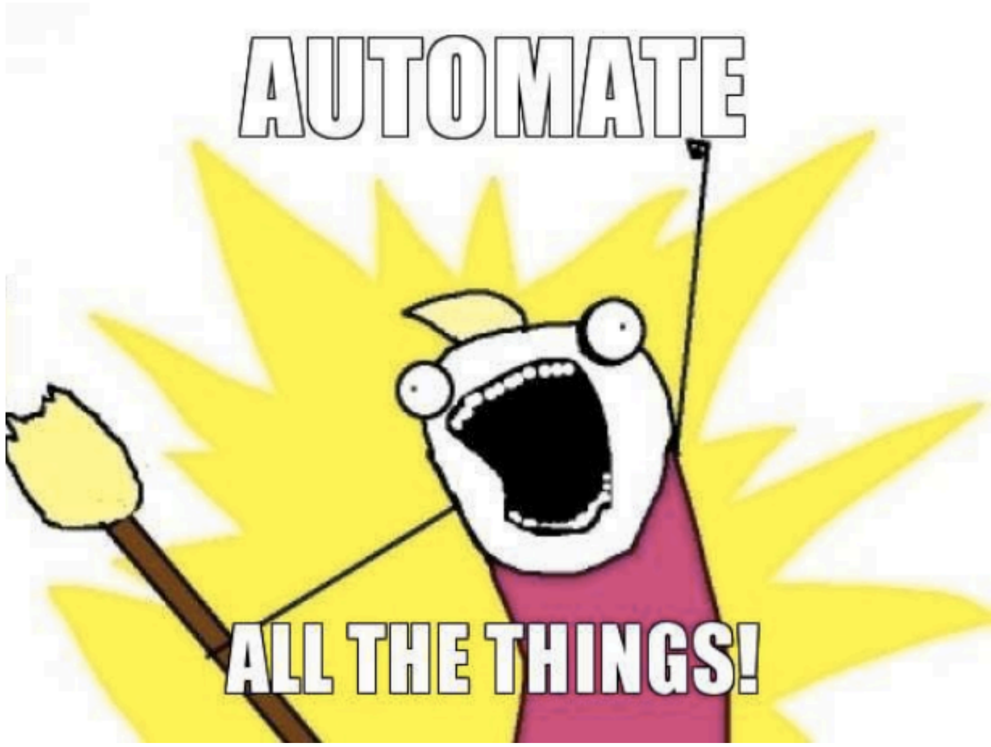
(These are mere means to reach the goal)

Provide Value

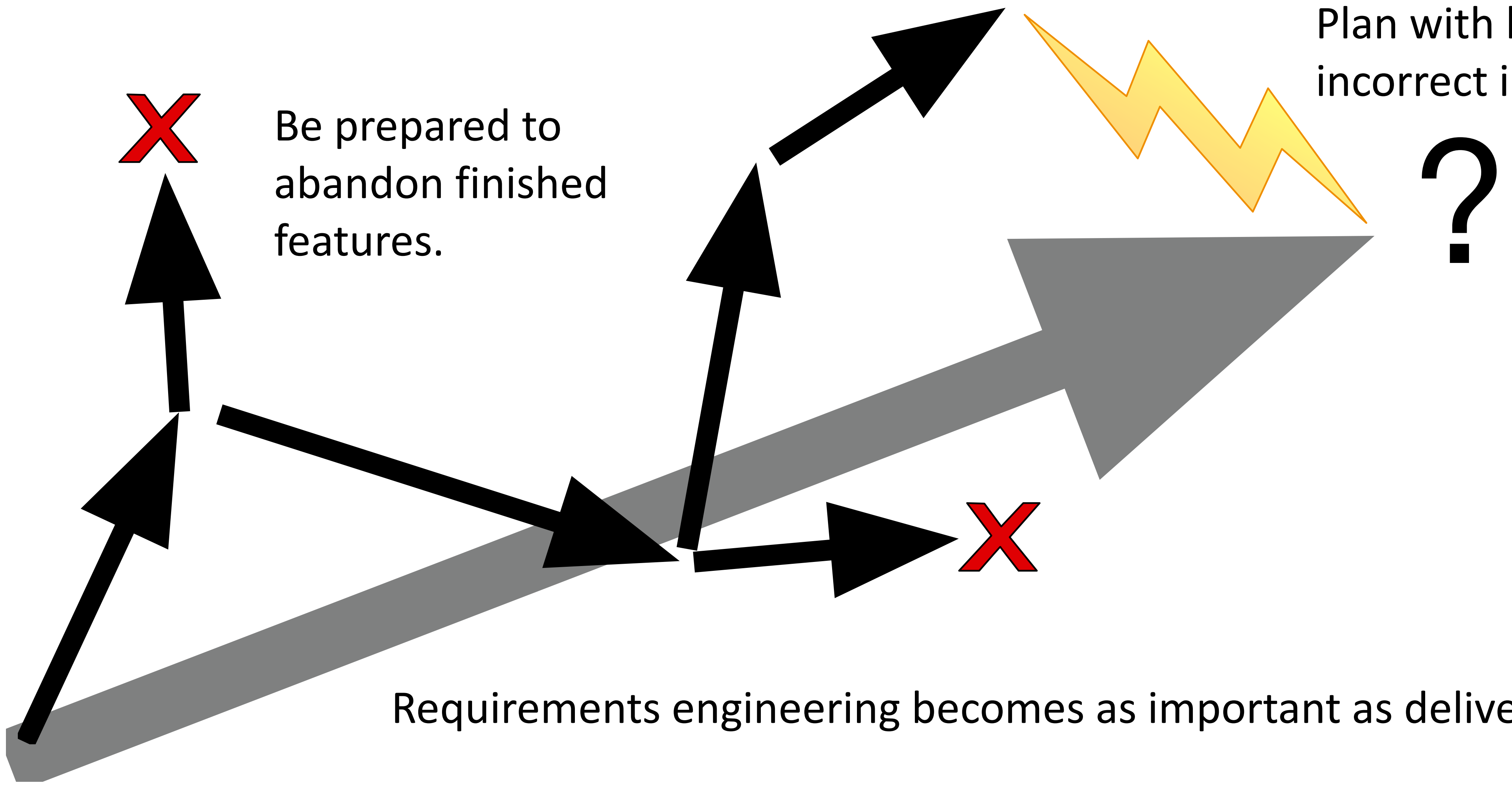
(Make something more efficient or create something new)

Cont. Delivery is Key for Value-Driven SE

- Reduced lead time means earlier **business impact**
- Accept that best solution is unknown up-front, **learn** and **improve** approach as you go
- Identify bad ideas earlier, so you can **try more ideas**



Experiment With Minimal Versions of Your Idea



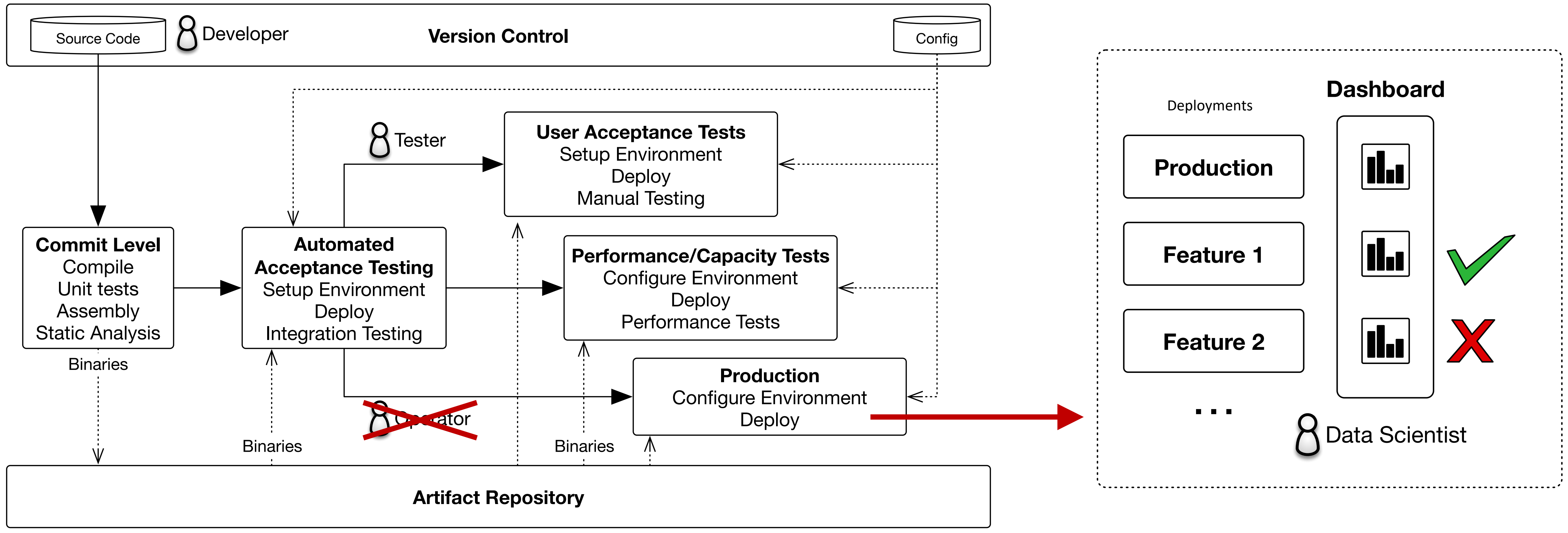
Be prepared to abandon finished features.

Plan with having an incorrect intuition

Requirements engineering becomes as important as delivering code

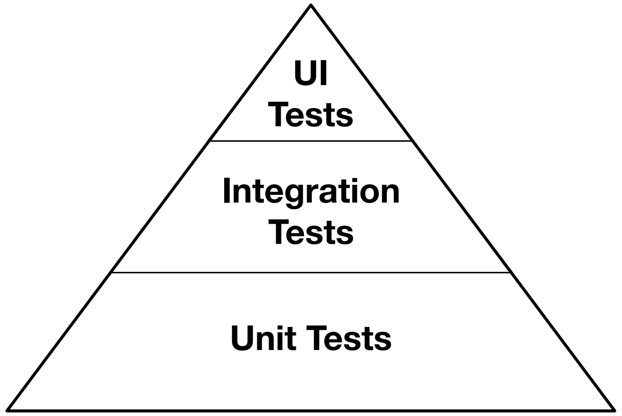
Continuous Experimentation

Continuous Experimentation Pipeline



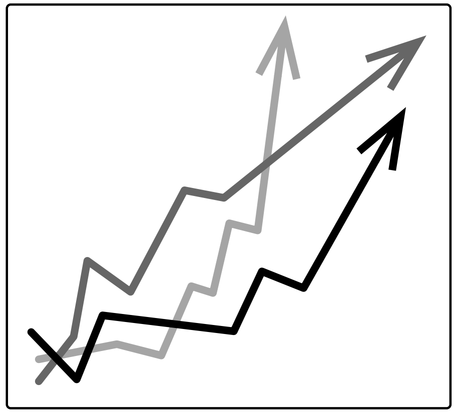
When To Continuously Experiment?

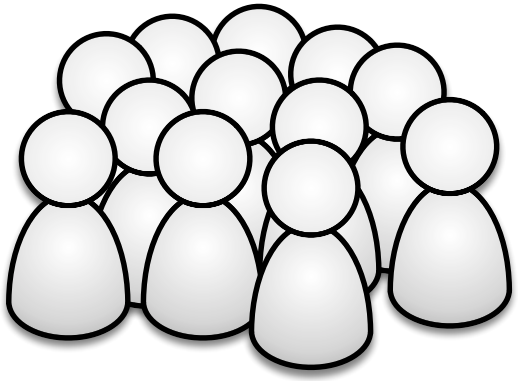
Does not replace traditional testing



Sometimes, scale is necessary for testing

(e.g., performance, machine learning, ...)

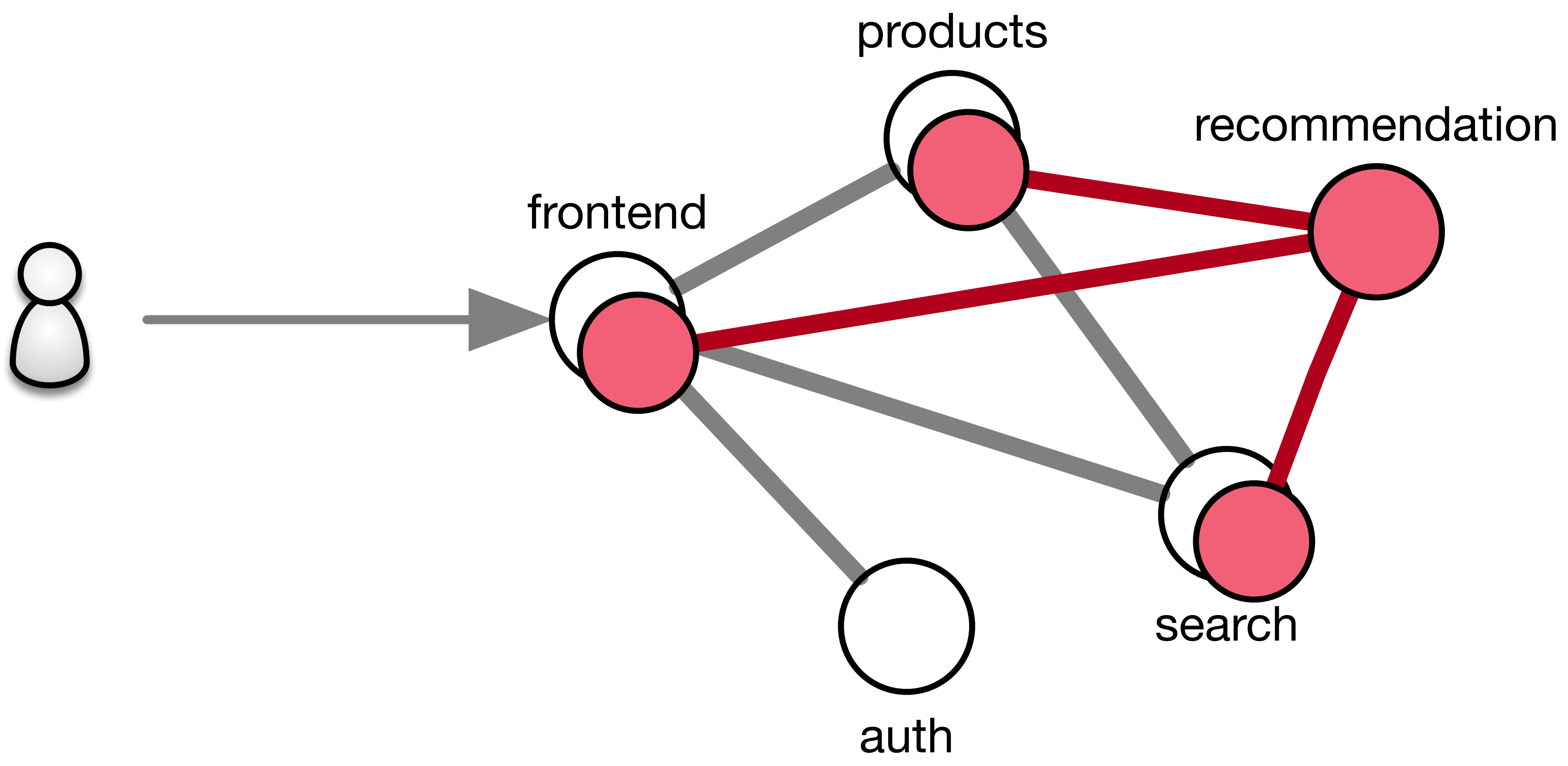




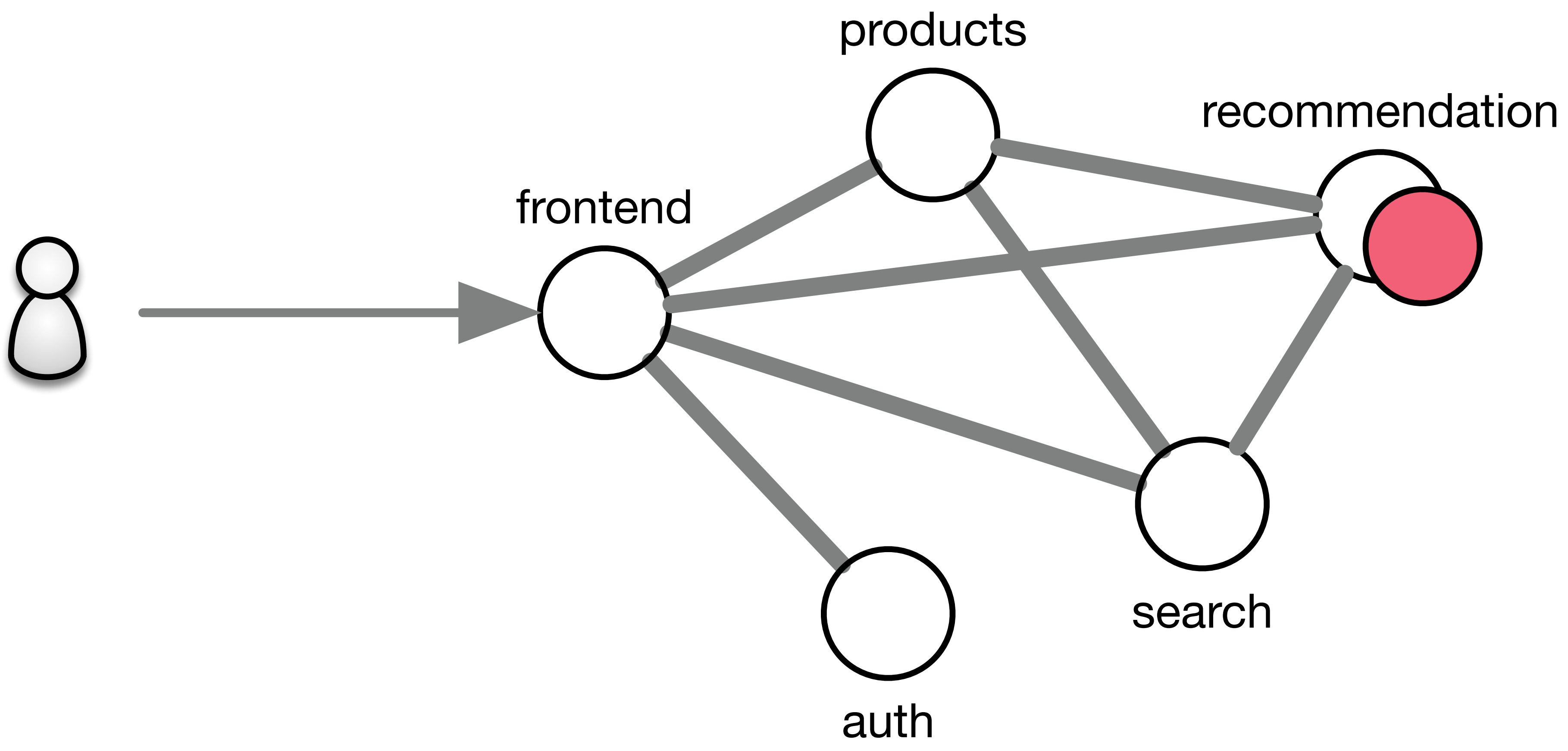
Humans... opinionated, irrational, have preferences.

How to experiment continuously?

Add a New Feature



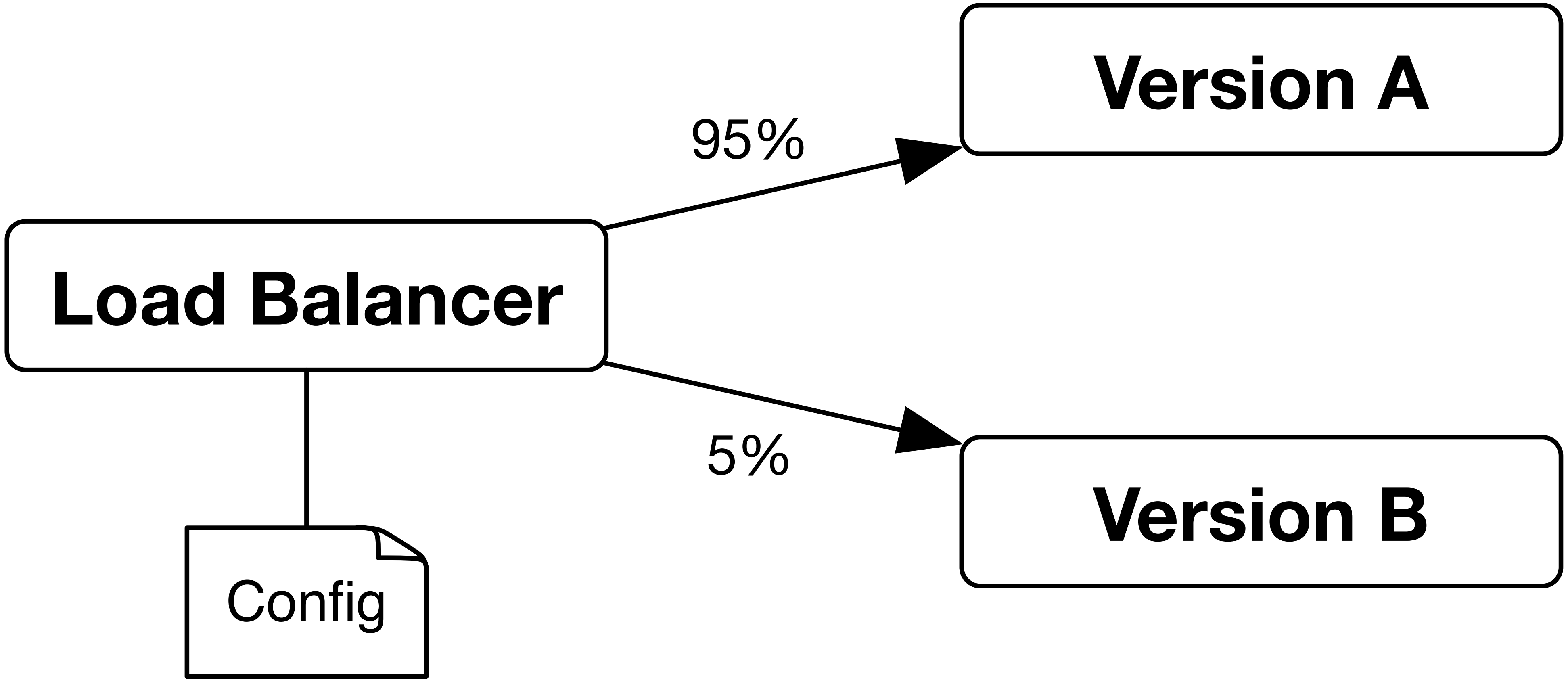
Add a New Version of a Feature



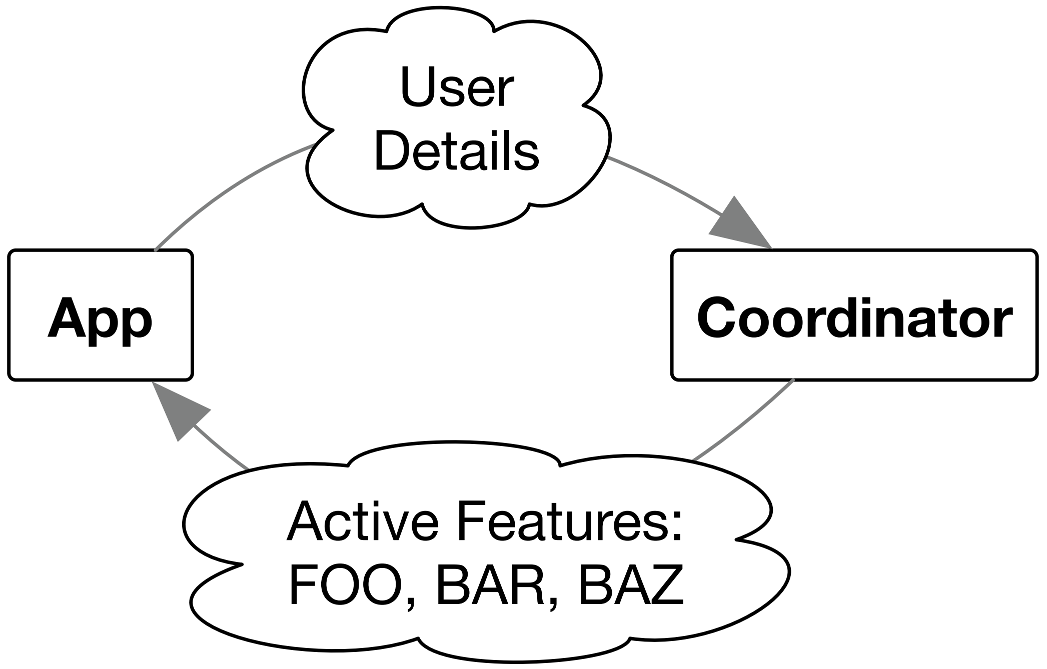
How to Setup an Experiment?

- Define experimental parameters
  - Participants (e.g., do not churn them)
  - Duration (e.g., day of week effect)
  - Hypothesis (e.g., falsifiable)
  - …
- Select relevant performance metrics
- Establish necessary infrastructure

Variant 1: Dynamic Traffic Routing



Variant 2: Feature Flags



```
if(isActive("FOO")) {
        ...
} else {
        ...
}
```

Interactions between active features present a maintainability nightmare. Keep features small, isolated, and short-lived.

Solution (Again) Depends on Use Case

Web Pages

Micro Services

Installed Software
(“Fat Clients”)

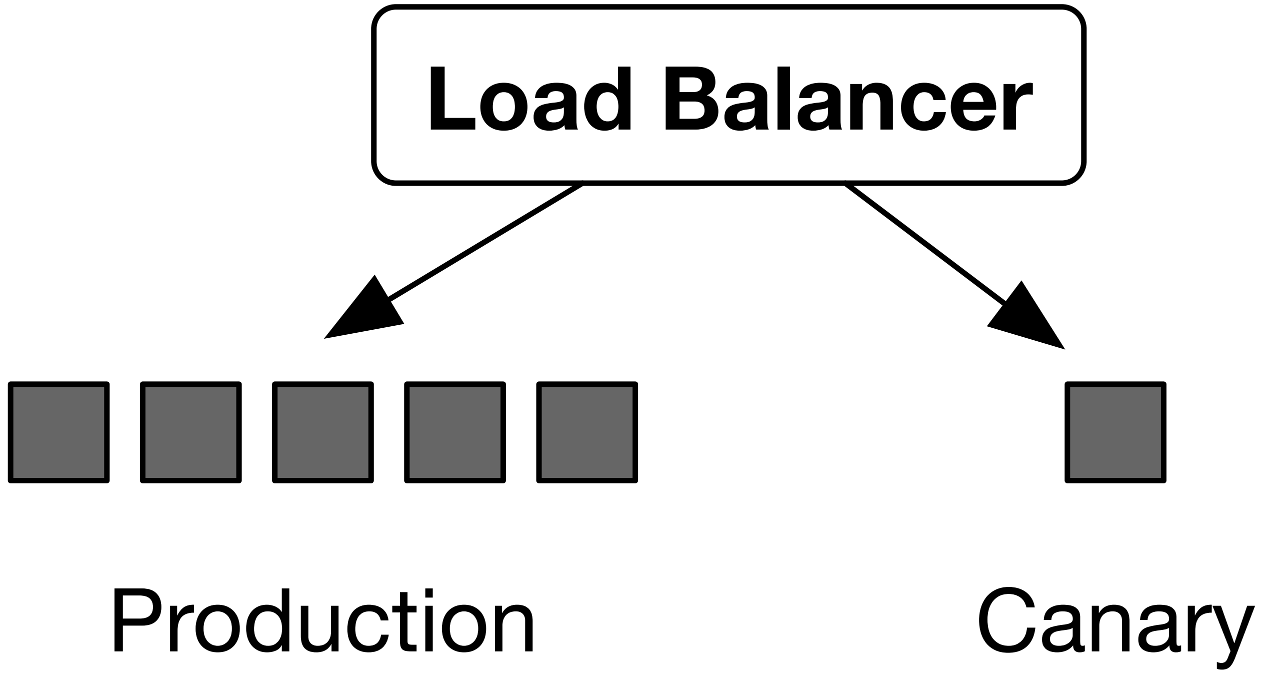
Mobile Phones
(App Stores)

Not an exhaustive list... Just food for thought.

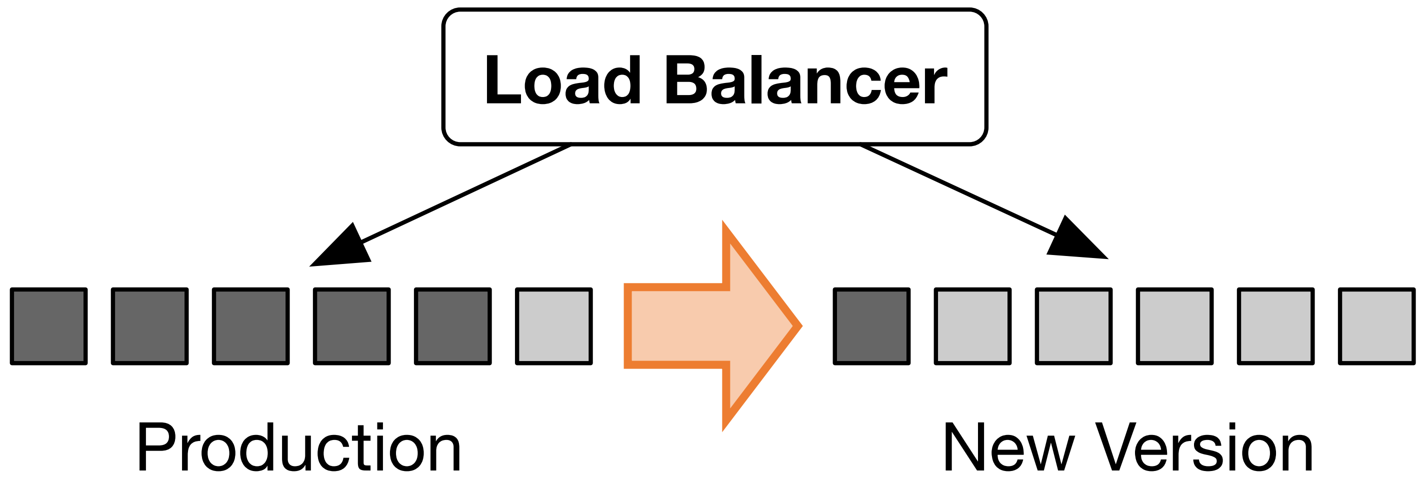
Implementation Strategies

Canary Release

- Internal Releases
  ("Eat your own dogfood")
- "Warm" audience
  (e.g., beta channel)
- "Cold" audience
  (e.g., random sample)



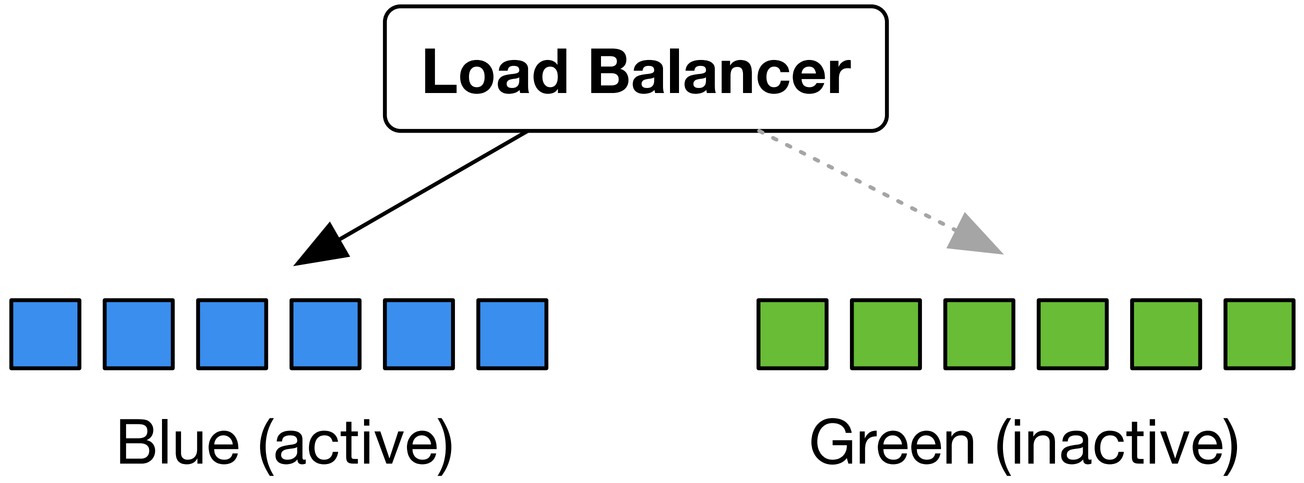
Gradual Rollout



With more confidence, serve more users with new version.

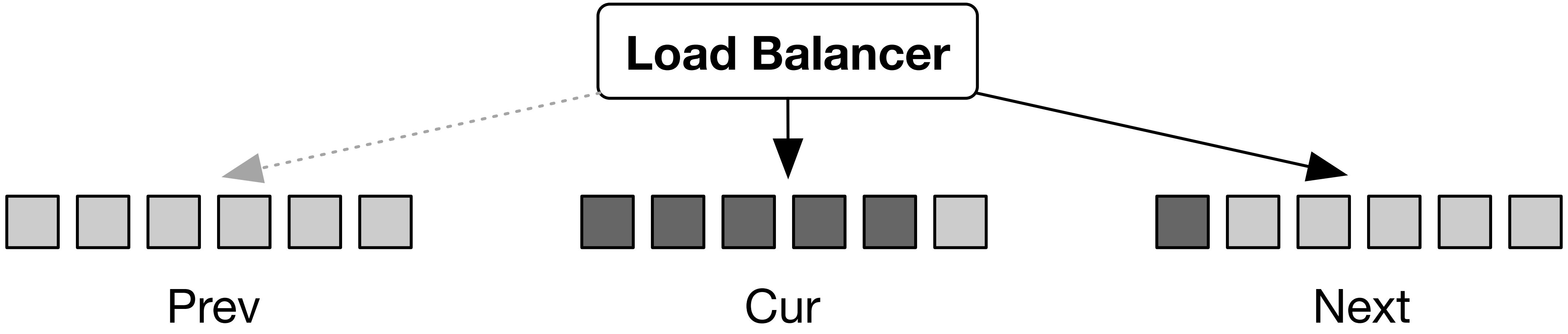
Blue/Green Deployment

- Host two identical environments
- One is active at a time
- Switch after new deployment
- Database migration needs automation



https://martinfowler.com/bliki/BlueGreenDeployment.html

Red/Black Deployment

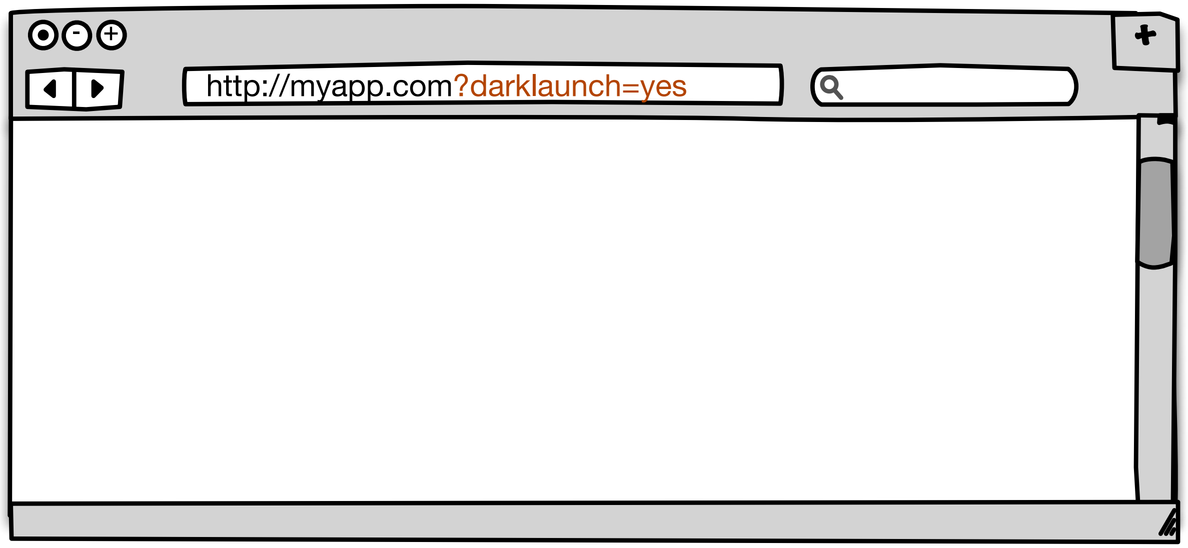


Use cloud instances to enable rolling updates

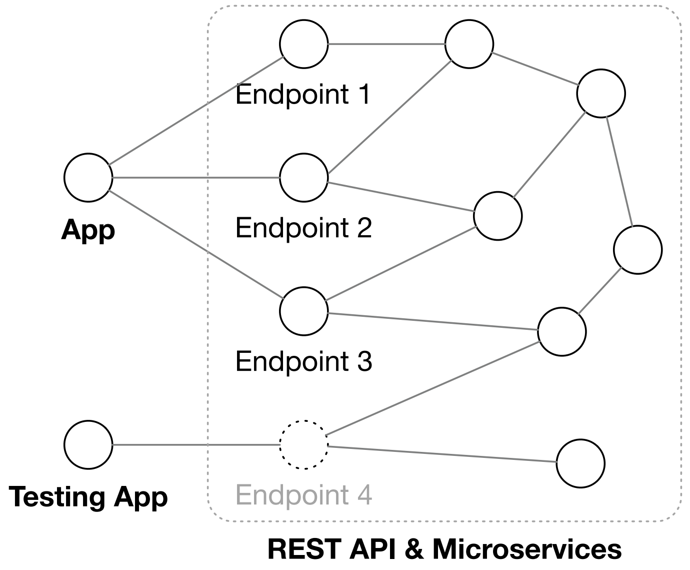
Dark Launch

Publish features that are not ready for release

Hidden Flags



Backend Launches

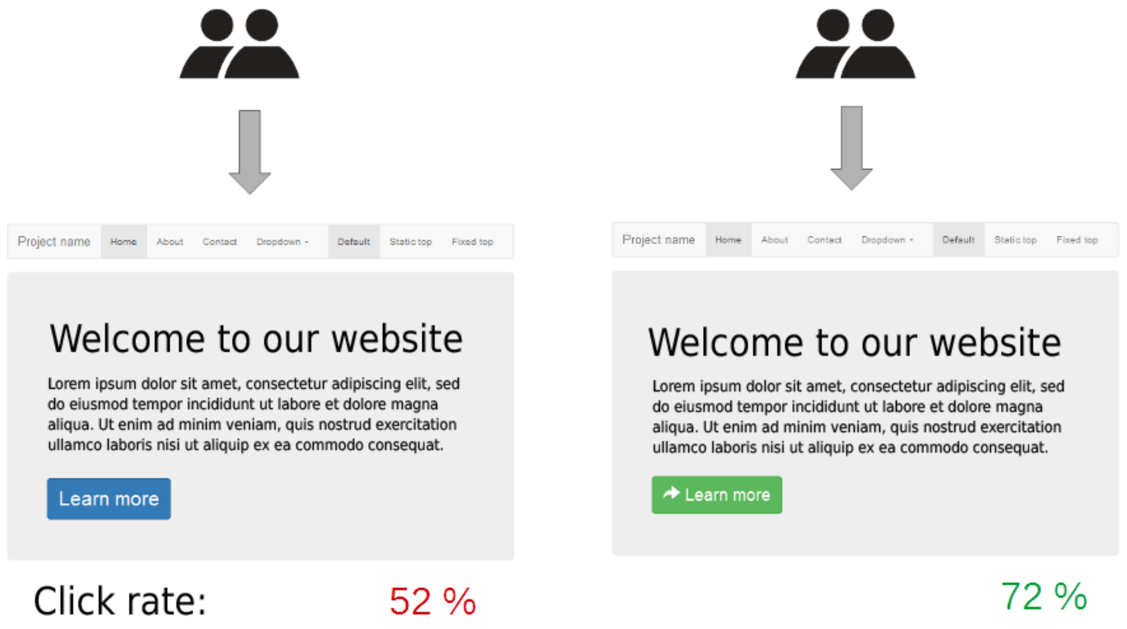


https://www.ibm.com/garage/method/practices/run/practice_dark_launch_feature_toggles/

So, how to decide..?

A/B Testing

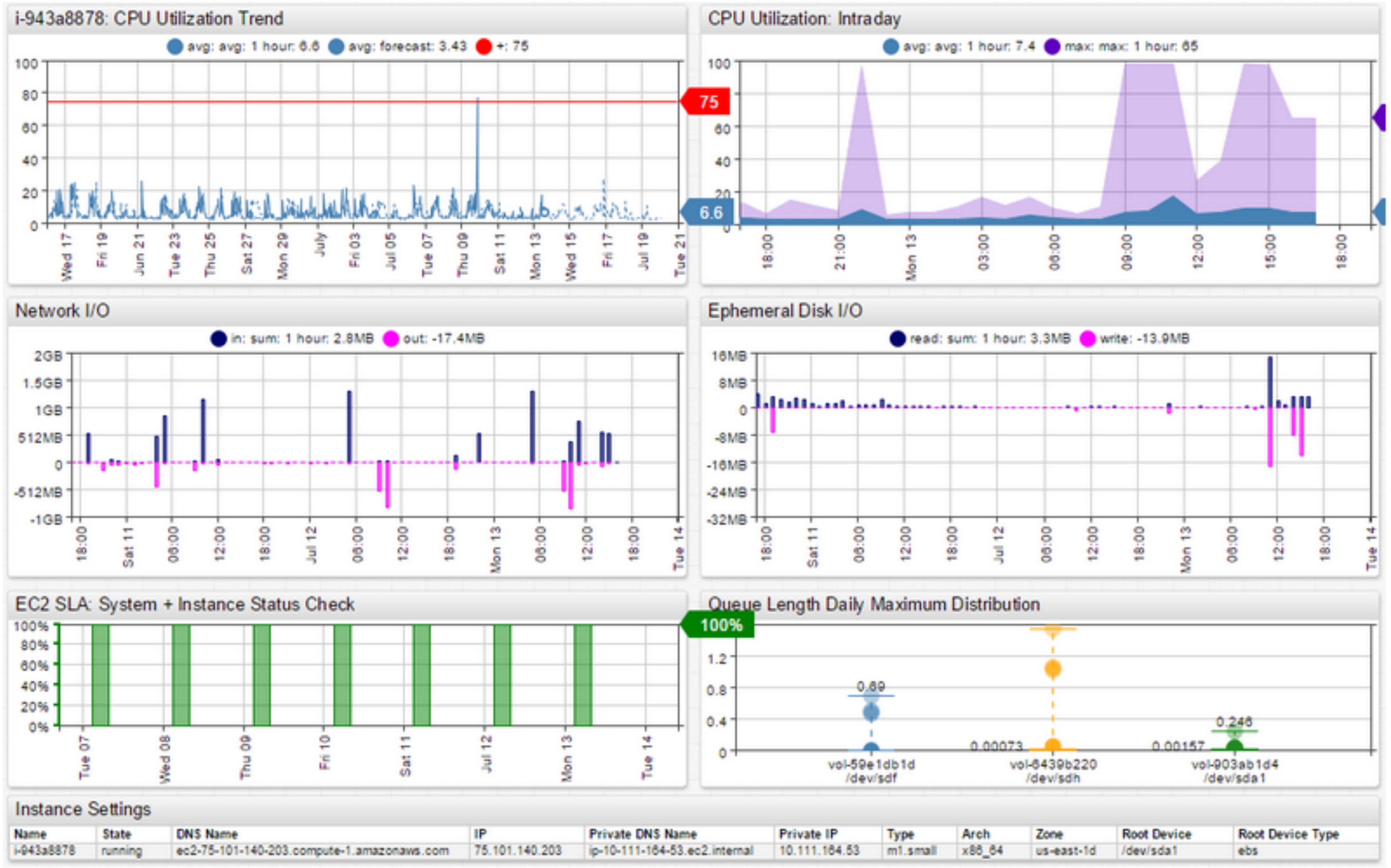
- Controlled experiment on two app versions
  - Production version (A)
  - One(!) new feature integrated into system (B)
- Proper metrics needed for comparison


Project name
Home
About
Contact
Dropdown
Default
Static top
Fixed top
Welcome to our website
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Learn more
Click rate:
52 %
72 %

https://upload.wikimedia.org/wikipedia/commons/2/2e/A-B_testing_example.png

General Performance Metrics

- CPU
- Memory
- Disk
  - Usage
  - IO
- Network IO

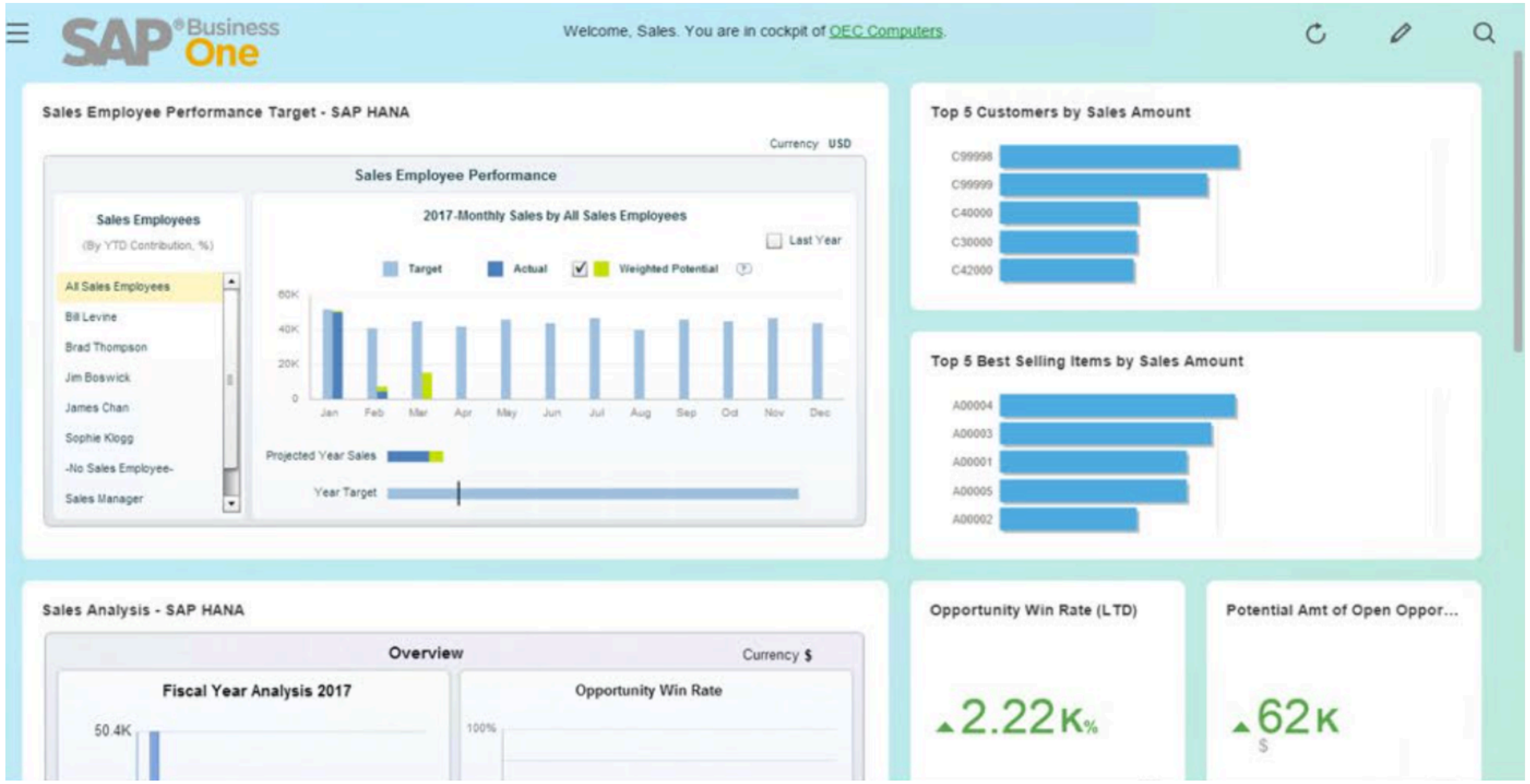


Instance Settings

Name	State	DNS Name	IP	Private DNS Name	Private IP	Type	Arch	Zone	Root Device	Root Device Type
i-943a8878	running	ec2-75-101-140-203.compute-1.amazonaws.com	75.101.140.203	ip-10-111-164-53.ec2.internal	10.111.164.53	m1.small	x86_64	us-east-1d	/dev/sda1	ebs

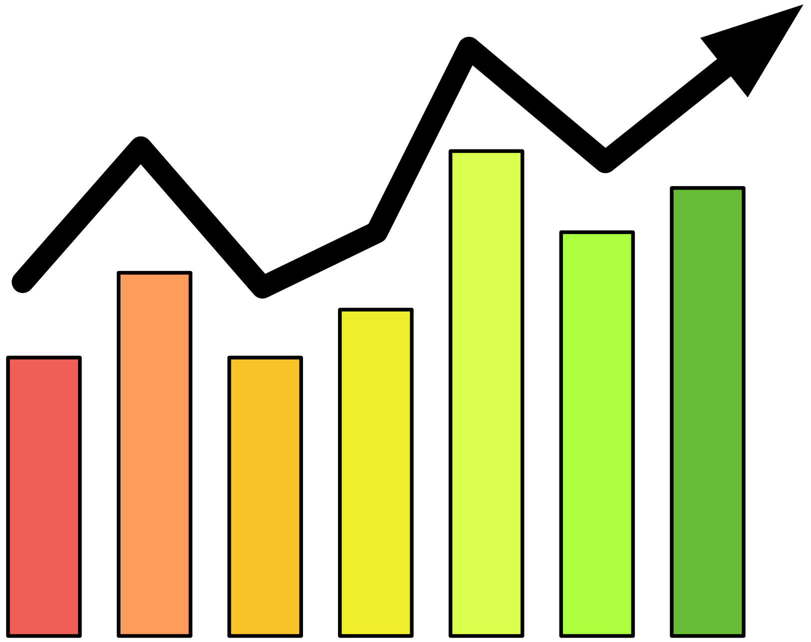
Business Metrics

- Sales / Revenue
- Costs
- New Users
- Loyalty
- Retention
- …



Domain / Experiment Specific

- Find relevant metrics
  - Click Rate
  - Navigation Paths
  - Number of Impressions
  - Time on site
  - …
- The metrics must help to (in-)validate the hypotheses



Monitoring vs. Logging

Logging

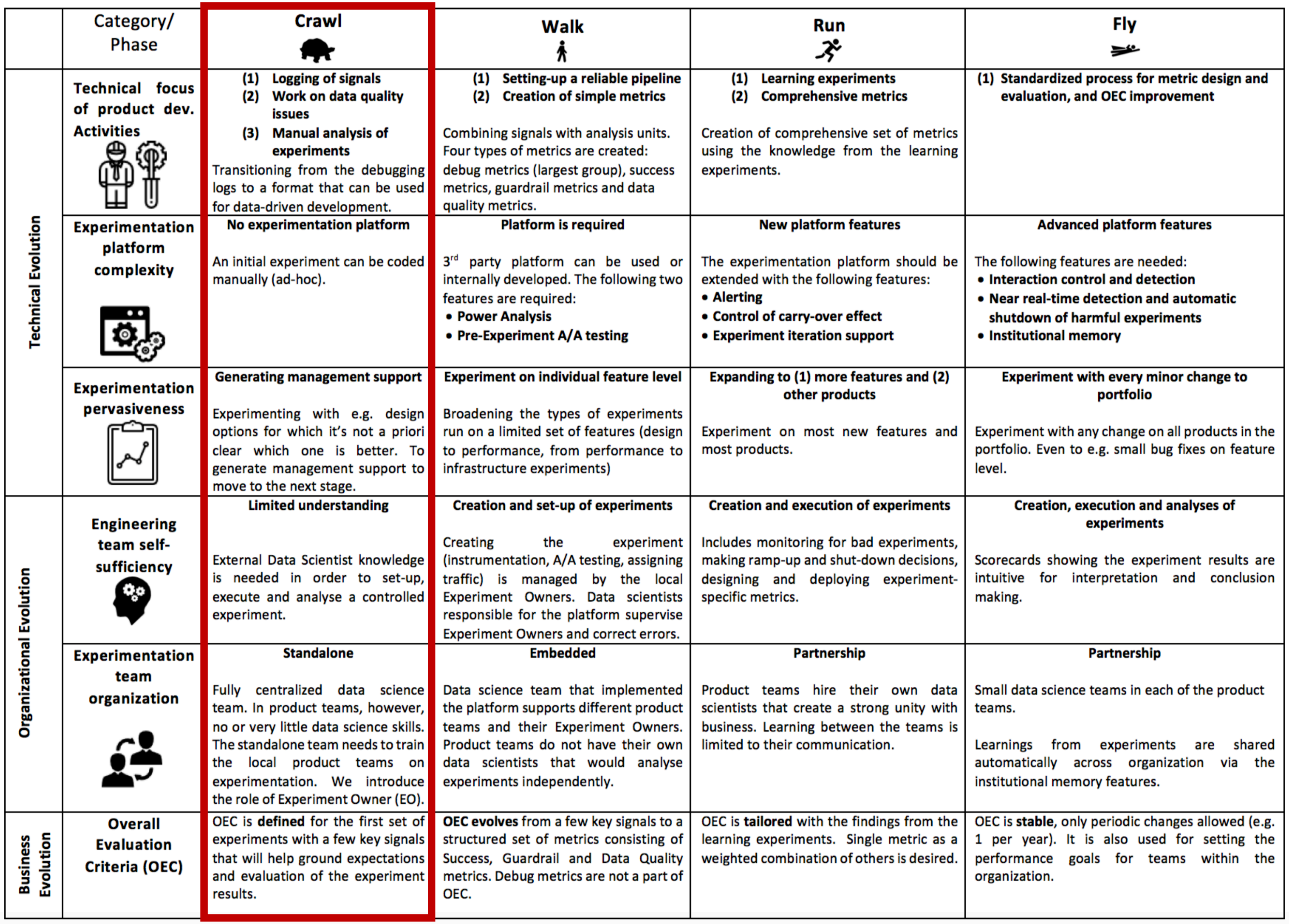
- Persist system events
- Often used for debugging or audits
- Low-level, raw data

Monitoring

- Collect and aggregate raw data
- Analyze metrics
- Generate insights

```
23:07:31.338334   Mail       com.apple.
23:07:31.338412   Mail       com.apple.m
23:07:31.338479   Mail       com.apple.m
23:07:31.338539   Mail       com.apple.me
23:07:49.850632   Google…    com.apple.mes
23:08:44.333718   Google…    com.apple.mes
23:09:54.677368   corebr…    com.apple.mess
23:09:54.677647   corebr…    com.apple.messa
23:10:06.633136   Google…    com.apple.messa
23:11:22.112721   eclipse    com.apple.messag
23:11:28.890316   Unknown    com.apple.message
23:11:34.400765   Google…    com.apple.message
23:12:12.835319   AOUMon…    com.apple.message.
23:13:09.049556   eclipse    com.apple.message.d
23:15:00.313393   eclipse    com.apple.message.d
23:16:12.126329   Dropbox    com.apple.message.do
23:16:12.126521   Dropbox    com.apple.message.dom
23:16:12.126577   Dropbox    com.apple.message.dom
23:16:12.126611   Dropbox    com.apple.message.doma
23:16:12.126637   Dropbox    com.apple.message.domai
23:16:12.126681   Dropbox    com.apple.message.domain
23:16:12.126725   Dropbox    com.apple.message.domain
23:16:23.665800   eclipse    com.apple.message.domain:
23:16:41.778358   Finder     com.apple.message.domain:
23:17:04.217796   eclipse    com.apple.message.domain:
23:20:18.103054   Dropbox    com.apple.message.domain: c
23:21:28.838100   Unknown    com.apple.message.domain: co
23:22:08.409639   eclipse    com.apple.message.domain: co
23:26:46.047709   Addres…    com.apple.message.domain: com
23:28:55.930930   eclipse    com.apple.message.domain: com.
23:30:31.369222   sudo       getgrouplist_2 called triggeri
23:31:28.898572   Unknown    com.apple.message.domain: com.a
23:32:15.065293   eclipse    com.apple.message.domain: com.ap
23:32:36.086980   Mail       com.apple.message.domain: com.ap
23:32:36.087048   Mail       com.apple.message.domain: com.app
23:32:36.087112   Mail       com.apple.message.domain: com.appl
23:32:36.087290   Mail       com.apple.message.domain: com.apple
23:32:36.087403   Mail       com.apple.message.domain: com.apple
23:32:36.087497   Mail       com.apple.message.domain: com.apple.
23:32:36.087568   Mail       com.apple.message.domain: com.apple.
23:35:22.854888   Dropbox    com.apple.message.domain: com.apple.c
23:35:22.854982   Dropbox    com.apple.message.domain: com.apple.co
23:35:22.855049   Dropbox    com.apple.message.domain: com.apple.cor
23:35:22.855100   Dropbox    com.apple.message.domain: com.apple.cor
23:35:22.855144   Dropbox    com.apple.message.domain: com.apple.core
```

The Rocky Road to Continuous Experimentation



	Category/ Phase	Crawl	Walk	Run	Fly
Technical Evolution	**Technical focus of product dev. Activities**	**(1) Logging of signals** **(2) Work on data quality issues** **(3) Manual analysis of experiments** Transitioning from the debugging logs to a format that can be used for data-driven development.	**(1) Setting-up a reliable pipeline** **(2) Creation of simple metrics** Combining signals with analysis units. Four types of metrics are created: debug metrics (largest group), success metrics, guardrail metrics and data quality metrics.	**(1) Learning experiments** **(2) Comprehensive metrics** Creation of comprehensive set of metrics using the knowledge from the learning experiments.	**(1) Standardized process for metric design and evaluation, and OEC improvement**
Technical Evolution	**Experimentation platform complexity**	**No experimentation platform** An initial experiment can be coded manually (ad-hoc).	**Platform is required** 3rd party platform can be used or internally developed. The following two features are required: • **Power Analysis** • **Pre-Experiment A/A testing**	**New platform features** The experimentation platform should be extended with the following features: • **Alerting** • **Control of carry-over effect** • **Experiment iteration support**	**Advanced platform features** The following features are needed: • **Interaction control and detection** • **Near real-time detection and automatic shutdown of harmful experiments** • **Institutional memory**
Technical Evolution	**Experimentation pervasiveness**	**Generating management support** Experimenting with e.g. design options for which it's not a priori clear which one is better. To generate management support to move to the next stage.	**Experiment on individual feature level** Broadening the types of experiments run on a limited set of features (design to performance, from performance to infrastructure experiments)	**Expanding to (1) more features and (2) other products** Experiment on most new features and most products.	**Experiment with every minor change to portfolio** Experiment with any change on all products in the portfolio. Even to e.g. small bug fixes on feature level.
Organizational Evolution	**Engineering team self-sufficiency**	**Limited understanding** External Data Scientist knowledge is needed in order to set-up, execute and analyse a controlled experiment.	**Creation and set-up of experiments** Creating the experiment (instrumentation, A/A testing, assigning traffic) is managed by the local Experiment Owners. Data scientists responsible for the platform supervise Experiment Owners and correct errors.	**Creation and execution of experiments** Includes monitoring for bad experiments, making ramp-up and shut-down decisions, designing and deploying experiment-specific metrics.	**Creation, execution and analyses of experiments** Scorecards showing the experiment results are intuitive for interpretation and conclusion making.
Organizational Evolution	**Experimentation team organization**	**Standalone** Fully centralized data science team. In product teams, however, no or very little data science skills. The standalone team needs to train the local product teams on experimentation. We introduce the role of Experiment Owner (EO).	**Embedded** Data science team that implemented the platform supports different product teams and their Experiment Owners. Product teams do not have their own data scientists that would analyse experiments independently.	**Partnership** Product teams hire their own data scientists that create a strong unity with business. Learning between the teams is limited to their communication.	**Partnership** Small data science teams in each of the product teams. Learnings from experiments are shared automatically across organization via the institutional memory features.
Business Evolution	**Overall Evaluation Criteria (OEC)**	OEC is **defined** for the first set of experiments with a few key signals that will help ground expectations and evaluation of the experiment results.	**OEC evolves** from a few key signals to a structured set of metrics consisting of Success, Guardrail and Data Quality metrics. Debug metrics are not a part of OEC.	OEC is **tailored** with the findings from the learning experiments. Single metric as a weighted combination of others is desired.	OEC is **stable**, only periodic changes allowed (e.g. 1 per year). It is also used for setting the performance goals for teams within the organization.

Fabijan et al. "The Evolution of Continuous Experimentation in Software Product Development", ICSE, 2017

Technical Focus of Product Dev. Activities

- Logging of Signals
- Work on data quality issues
- Manual analysis of experiments

Transitioning from the debugging logs to a format that can be used for data-driven development

Experimentation Platform Complexity

- No experimentation platform

An initial experiment can be coded manually (ad hoc)

Experimentation Pervasiveness

- Generating Management Support

Experimenting with, e.g., design options, for which it is not a priori clear which one is better. To generate management support to move to the next stage.

Engineering Team Self-Sufficiency

- Limited understanding

External Data Scientist knowledge is needed in order to set-up execute and analyze the controlled experiment.

Experimentation Team Organization

- Standalone

Fully centralized data science team. In product teams, however, no or very little data science skills. The standalone team needs to train the local product teams on experimentation. We introduce the role of Experiment Owner.

Overall Evaluation Criteria

OEC is defined for the first set of experiments with a few key signals that will help ground expectations and evaluation of te experiment results.

Conclusion

Interesting Connection Points

- DevOps
- Release Engineering in Practice
  - Site Reliability Engineering
  - Build Sheriff
  - …
- Data Science
- Software Product Line

After today's lecture, you...

- can explain rationale behind CE
- know how to technically enable CE
- can explain the differences between various implementation strategies
- understand the role of monitoring and A/B testing
- know about the evolution of CE in a project