

Achieving 99.99% uptime

A tale of Observability

Alex Bicalho

Autodesk Forge - Run Platform Area Lead

FORGE DEVCON





About the speaker

Alex Bicalho

Alex has been working at Autodesk for 19 years. He worked as a QA Engineer in 3dsmax, AutoCAD and the Platforms Graphics SDK, later in the Cloud Platform team as a Test Engineering Manager, as an Infrastructure Engineering Manager as Product Manager for Observability, and most recently leading the Forge Run Platform Area in the Forge team. Alex believes in Automating all repetitive tasks; in building resilient, cost effective and scalable systems, and in delivering high quality services to our customers

Observability



Observability

Observability as an SRE discipline implies the ability to effectively detect, diagnose and analyze based on

***outputs** produced by a system*

Four Pillars of Observability

Monitoring

Alerting/Visualization

Distributed Tracing

Log Aggregation/Analytics



Photo by Lucas Pezeta

The Cloud at Autodesk

DIFFERENT OPERATING SYSTEMS

- Windows Server
- Linux

DIFFERENT PROGRAMMING LANGUAGES

- Java
- Scala
- .NET on Windows
- .NET on Linux (Mono)
- Python
- Ruby
- NodeJS
- Go

DIFFERENT IMPLEMENTATIONS

- VM on Datacenter
- VM on AWS EC2
- Containers on AWS ECS
- Containers on AWS Fargate
- Containers on Kubernetes
- AWS Lambdas

Observability Numbers

19

TOOLS

Multiple Observability
tools in early 2019

20%

DETECTION

20% of the incidents
were detected by our
monitoring

1%

SIGNAL TO NOISE

Most of our events and
alerts are noise, they're
not actionable

Strategy



Reliability is the most important **feature** of any application or system. Customers will not be able to take advantage of **new features** if they **are not reliable**.

Research

Market:

- Market trends
- Conference Papers
- Vendor research and interviews

Books, Articles:

- Google SRE Book
- DORA Metrics
- Coursera SLI/SLO training

New Observability Strategy

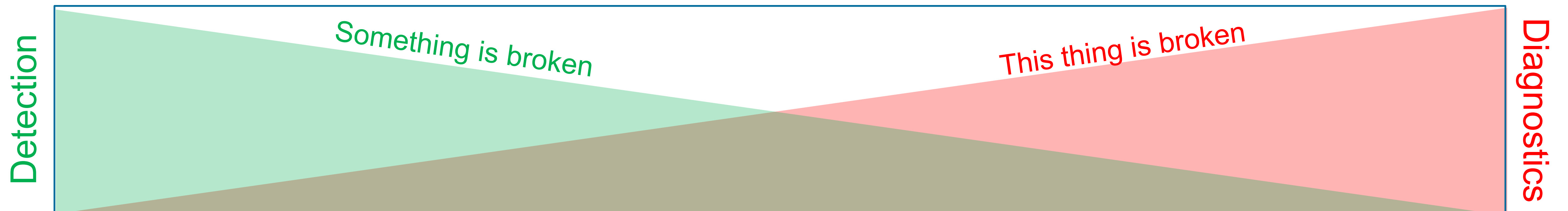
3 Lines of Defense

- ### 1 SLI/SLO/ERROR BUDGETS

 - Based on Request & Response
 - Measured closest to customer
 - Error Budget triggers **Alerts**
- ### 2 SYNTHETIC TESTS

 - Main Workflows for SLO
 - SLO workflows trigger **Alerts**
 - Others generate **Tickets**
- ### 3 CUSTOM METRICS

 - Focused on Diagnostics
 - Include Logging & other tools
 - Trigger **Tickets**



Terminology

SLI

Quantitative measure of some aspect of the level of service



SLI

Expressed as **Good Events/Valid Events**. Measure as close to the user as possible.

e.g. request latency, error rate

SLO

Target value or range for a service level that is measured by an SLI



SLO

Represents the line between happy and sad users. A good SLO has predictable relationship with user happiness.

e.g. 99.95% success rate in 28 days

ERROR BUDGET

Common incentive for Product Management and Engineering to find the right balance of investment

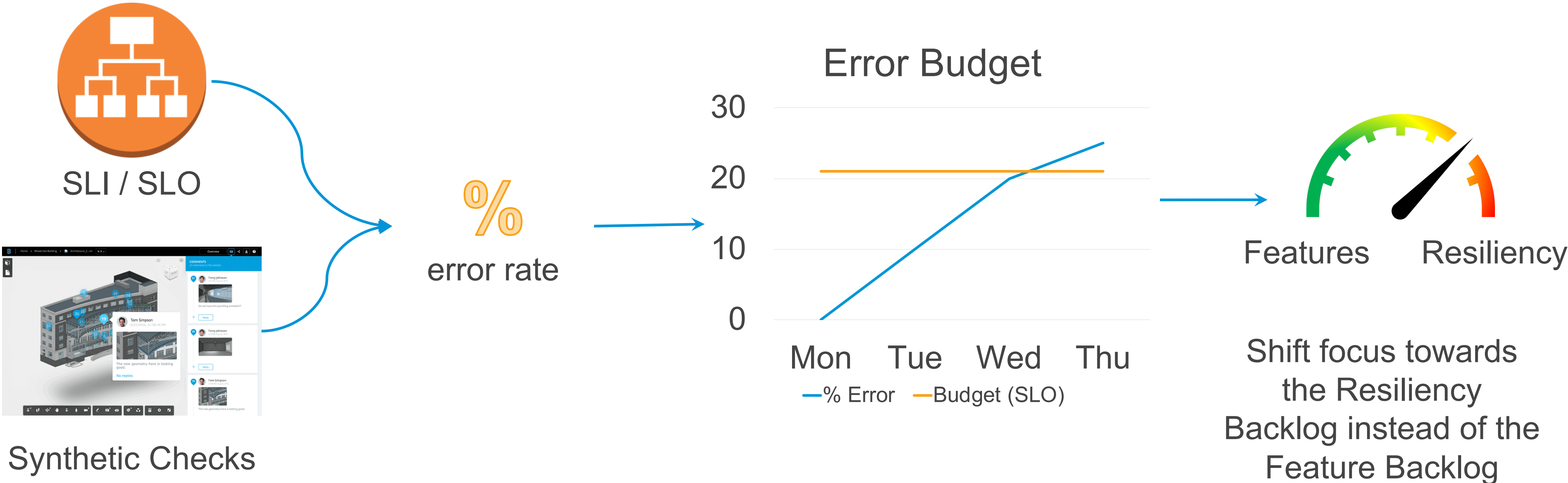


Error Budgets

Inverse of SLO. Calculated over a **28 Day** Period. **Alerts** are triggered when x% of budget is consumed within a time window.

e.g. for 99.95% SLO, the error budget is 0.05%.

Customer Happiness



Implementation



Implementation

Measured at the Load Balancer – closest to the **customer**
computed once per minute

SLI:

- SLI for Error Rates (HTTP 500)
- SLI for Latency (P99, P50, etc)
- SLI measured for different/individual API paths

SLO:

- Specifies the threshold for success rate -> ex.: 99.9% success rate
- Specifies latency thresholds:
 - P99 $\geq$ 300ms
 - P50 $\geq$ 1000ms

Implementation

ERROR RATE

$$1 - \frac{(\text{Total Calls} - \text{Number of HTTP 5xx})}{\text{Total Calls}}$$

P99, P50

P(99)Latency values

or

P(50)Latency values

COMPUTE ERROR BUDGET OVER TIME

Given a 28 days window, the budget is as follows:

Target Availability	Error Budget
99.9%	40m 19.2s
99.95%	20m 9.6s
99.99%	4m 2s

ERROR BUDGET CONSUMPTION

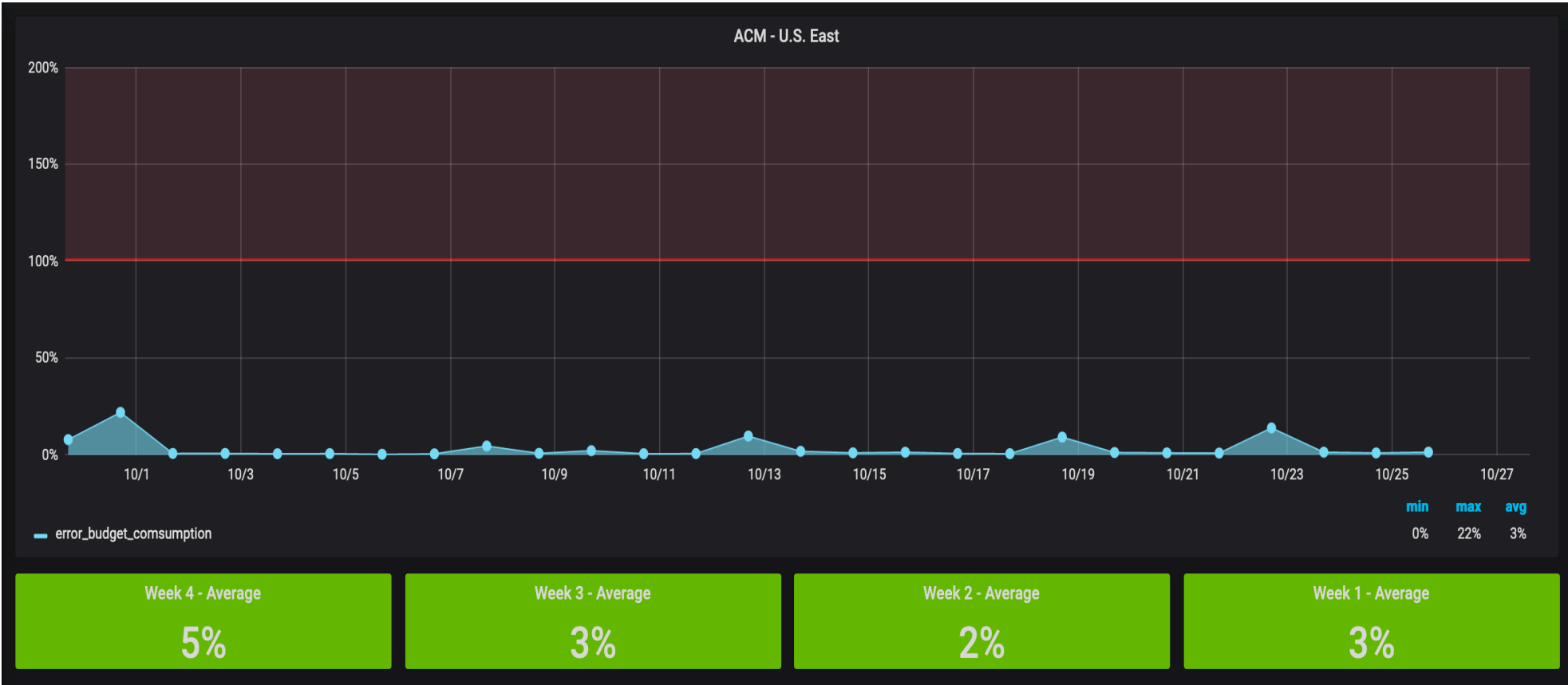
Compute the Pass/Fail over the Error Budget period

Alert based on:

- Speed of Error Budget consumption
- % threshold of Consumption

Visualization

SUMMARY DASHBOARD



DETAILED DASHBOARD



Synthetic Checks

SIMULATE CUSTOMER USAGE

- Run every few minutes
- Exercise a Customer workflow or an API call that represents a customer's journey through the service

PROS

- Represents a customer workflow
- Will highlight issues that happen with inter-service dependencies

CONS

- Cannot run frequently
- Can be 'flaky' or flap between pass/fail
- Will not necessarily reflect a customer's point of view
Suffers from 'observer bias'

Logging Policies

- Log at the proper level (INFO vs WARN vs ERROR)
- Use Feature Flags to change Log Level in production
- Use structured, machine readable, searchable logs
- Create a standardized logging schema for your company, or use a common schema, like Open
- Log intentionally, not as `printf()`
- Do not use logging for Analytics or Metrics

Summary



Where are we now?

6

TOOLS

Down from 19 to 6 tools and services, and we are migrating/deprecating the old ones

52+

SLI/SLO

We onboarded 52+ apps and services into SLI, SLO and Error Budget

3

STRATEGIES

We have an Observability Strategy, followed by Logging and Distributed Tracing strategies

400+

AUTODESK
EMPLOYEES TRAINED

We provided training about SLI, SLO and Error Budgets to all of Forge Engineering

Where are we now?

CUSTOMER CENTRICITY

We know when customers are being impacted, which application or service has issues

STACK SIMPLIFICATION

We are reducing noise, and becoming intentional with what's monitored, measured, reported

COST SAVINGS

We have less tools, less maintenance, more automation, and we are not changing tools frequently

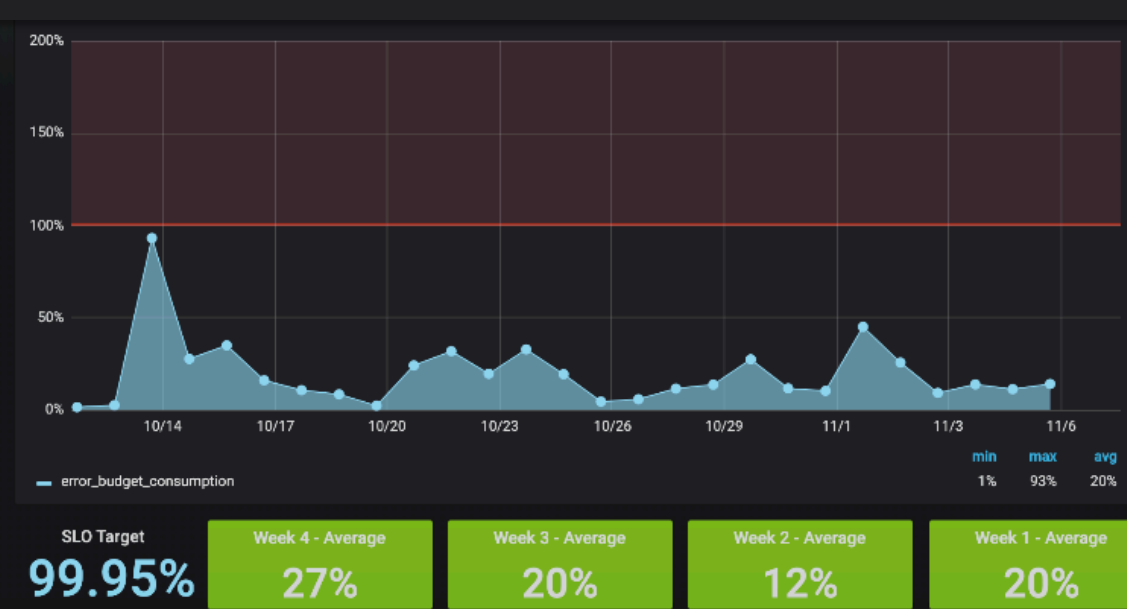
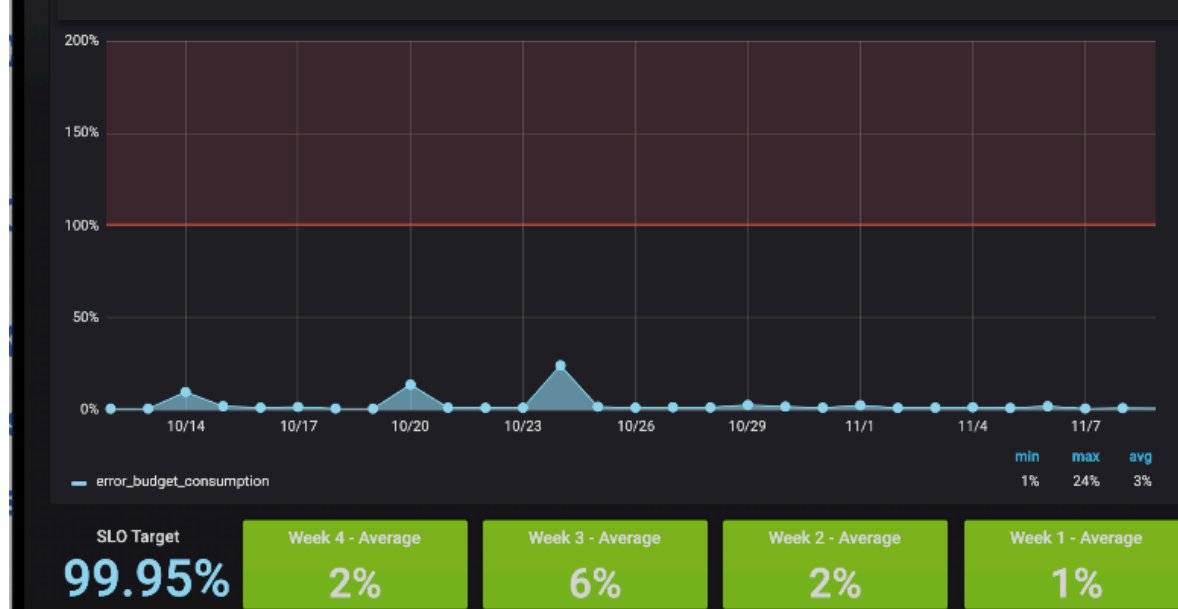
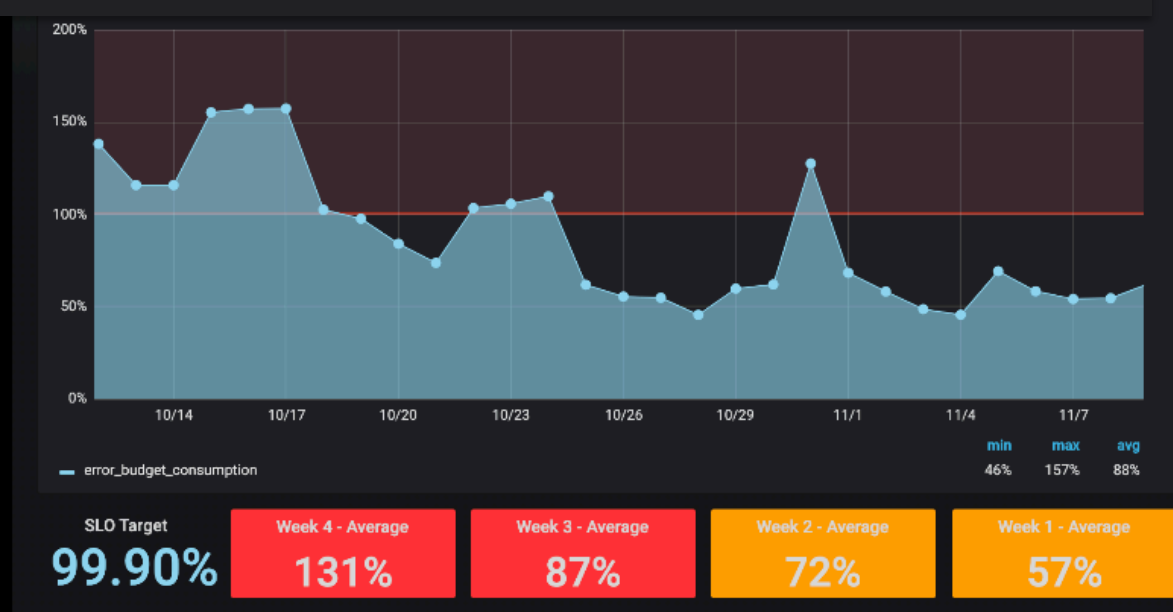
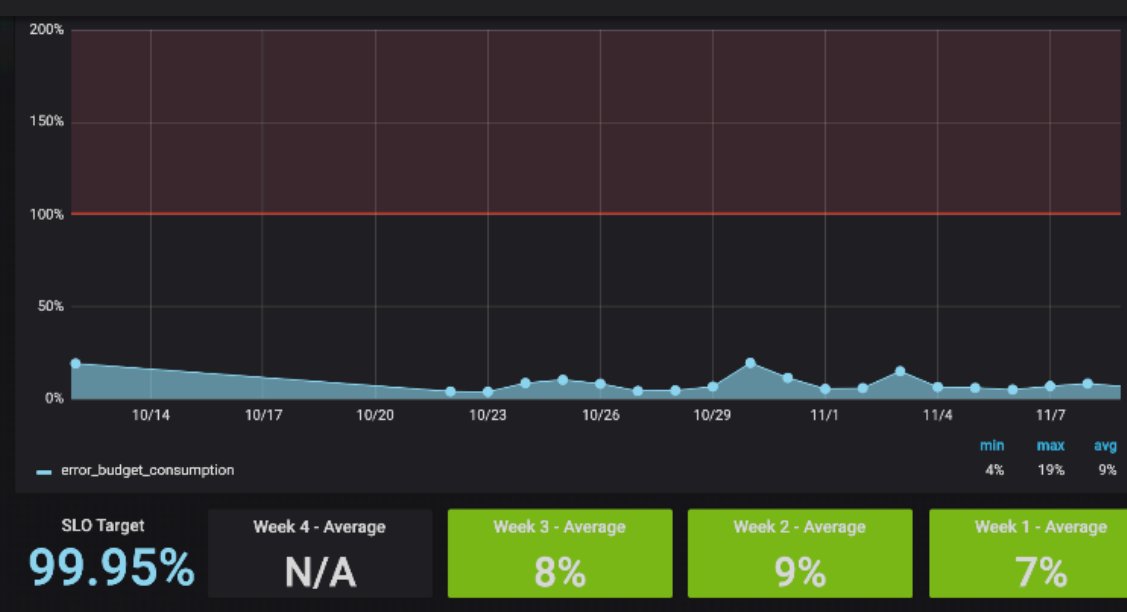
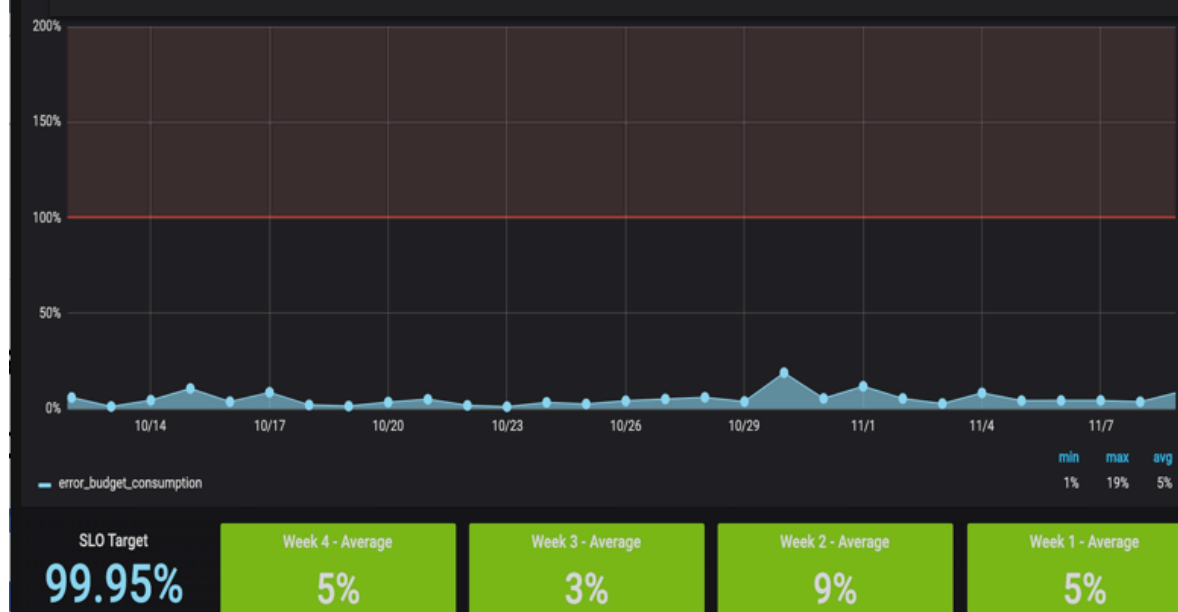
DRIVING GOOD BEHAVIOR

Several teams have started taking action to fix issues based on the Error Budget dashboards

Where are we now?



28 Day Error Budget Consumption



We welcome
your feedback

What can you do?

SLI, SLO, ERROR BUDGET

Implement error budgets:

- Start by measuring simple SLIs
- Define SLOs and iterate
- Create Error Budget dashboards
- Socialize an Error Budget Policy

SYNTHETICS & CUSTOM METRICS

Synthetic Checks:

- Create checks as SLO:
 - Login and Launch
 - Basic Functionality

Custom Metrics:

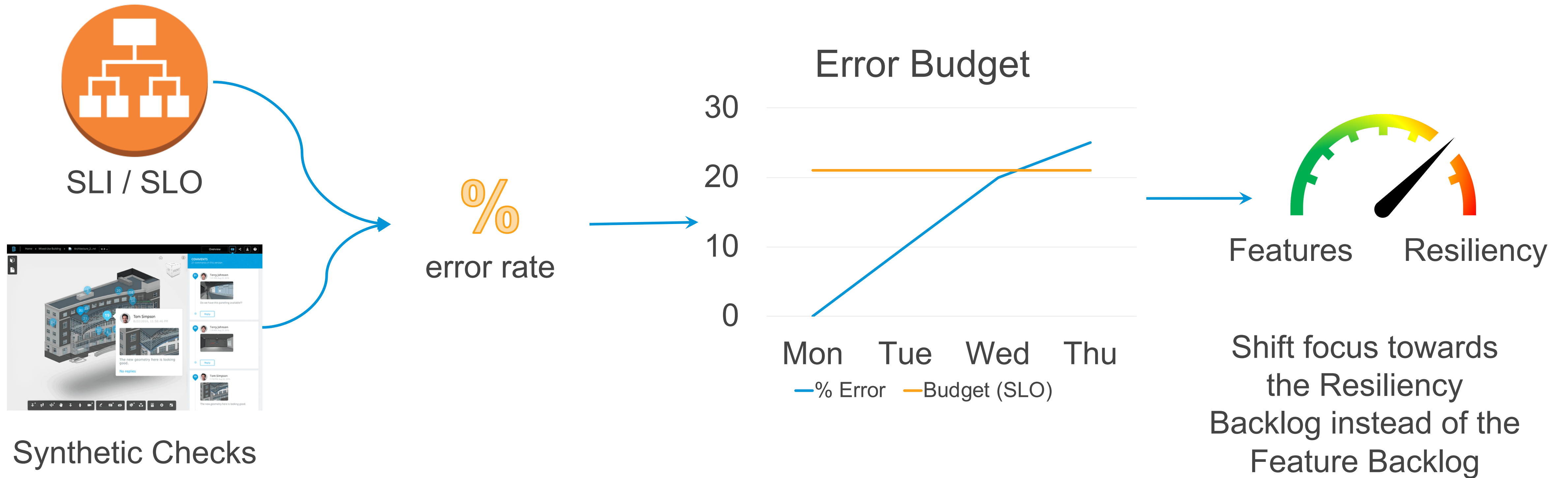
- Setup System Metrics, without alerting, but with intentionality
- Setup Analytics metrics

LOGGING

- Get rid of Printf() Logging!
- Setup a Schema
- Setup Policies

Reliability is the most important **feature** of any application or system. Customers will not be able to take advantage of **new features** if they **are not reliable**.

Customer Happiness



Q&A





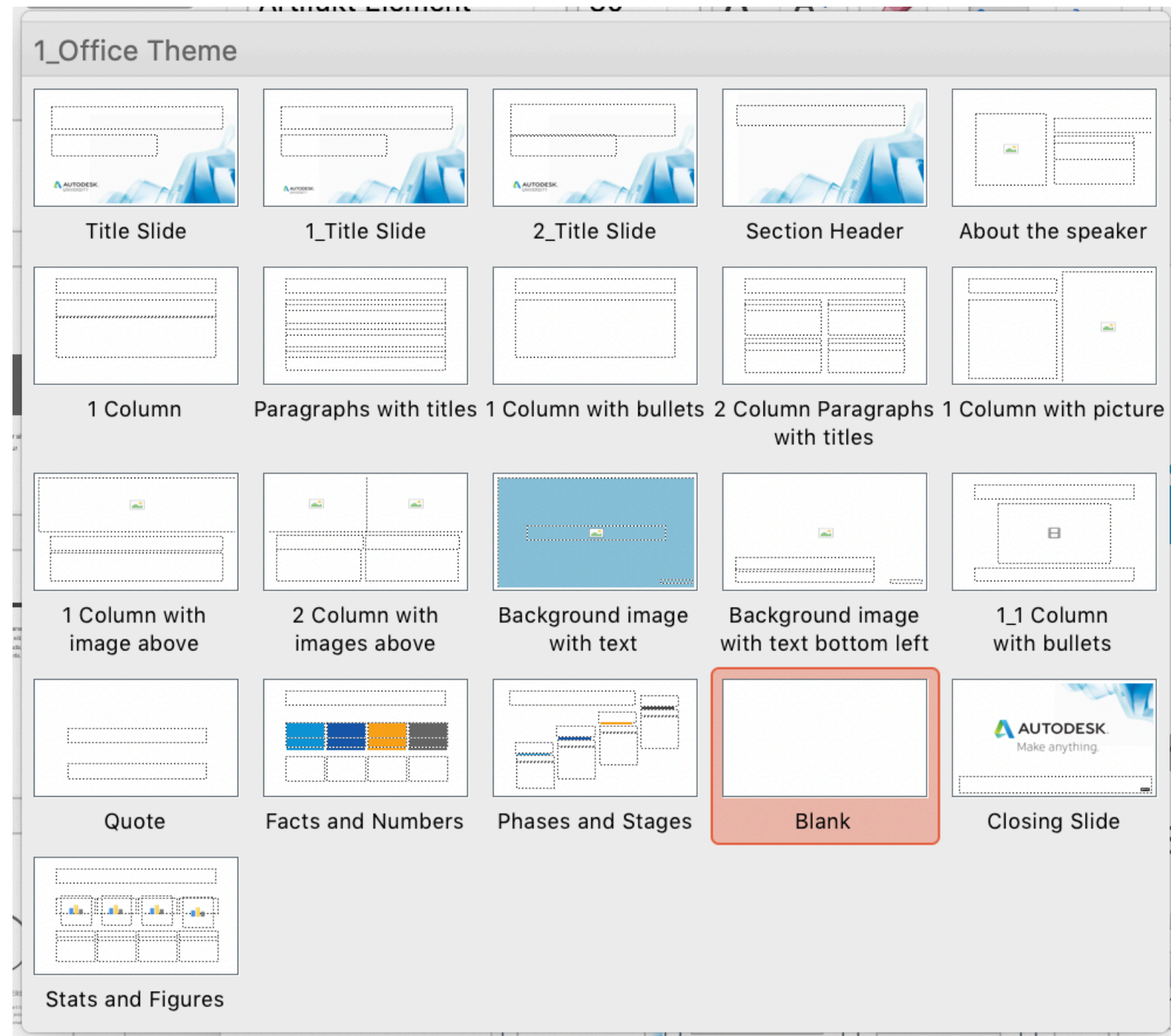
Autodesk and the Autodesk logo are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product and services offerings, and specifications and pricing at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document.

© 2019 Autodesk. All rights reserved.



Instructions and resources (Delete page)

- Access the slide layouts for this template on the Home tab under Slides/Layout.



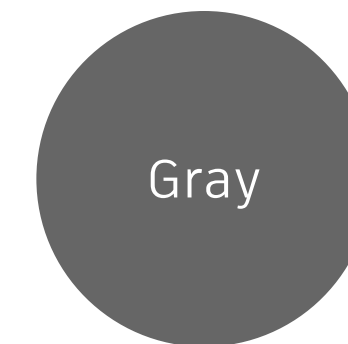
- AU 2019 Fonts: Artifakt or as a backup please use Arial.

- AU 2019 Colors



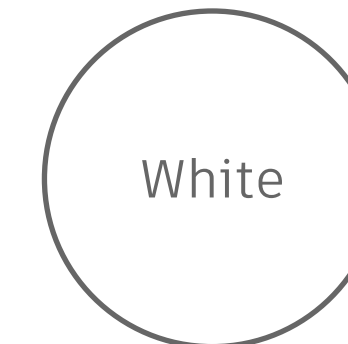
ADSK
blue

R 6 G 150 B 215
HEX: 0696D7



Gray

R 102 G 102 B 102
HEX: 666666



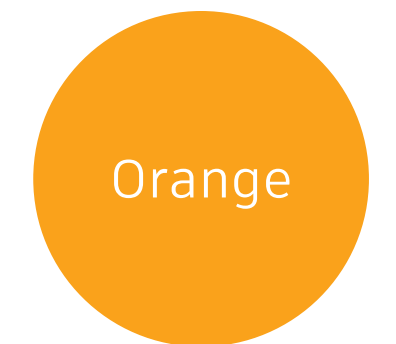
White

R 255 G 255 B 255
HEX: FFFFFFFF



Dark
blue

R 24 G 88 B 168
HEX: 1858A8



Orange

R 250 G 162 B 27
HEX: FAA21B

- AU Resources
 - Branding and editorial guidelines – Autodesk [Brand Hub](#)