Config/Infrastructure as Code: 2.0?

Pulumi

$ whoami

- Marcel Kornegoor
- Co-owner & head of training @ AT Computing
- Cloud, DevOps & Continuous Delivery fanatic
- Public speaker
- Bad joke junky
- And yes, I've become an open source nerd

ToC

- Root cause
- The observed trend & the problem
- <nerd stuff>
- WingLang
- Crossplane
- Pulumi
- Pkl
- #rant

Root cause – why am I here?

- Attending Config Management Camp
- Pondering about abstraction levels
- Thinking about the role of AT Computing
- Lots of networking with peers
- Fiddling around with new tools

The perceived problem

Everything is too complex.

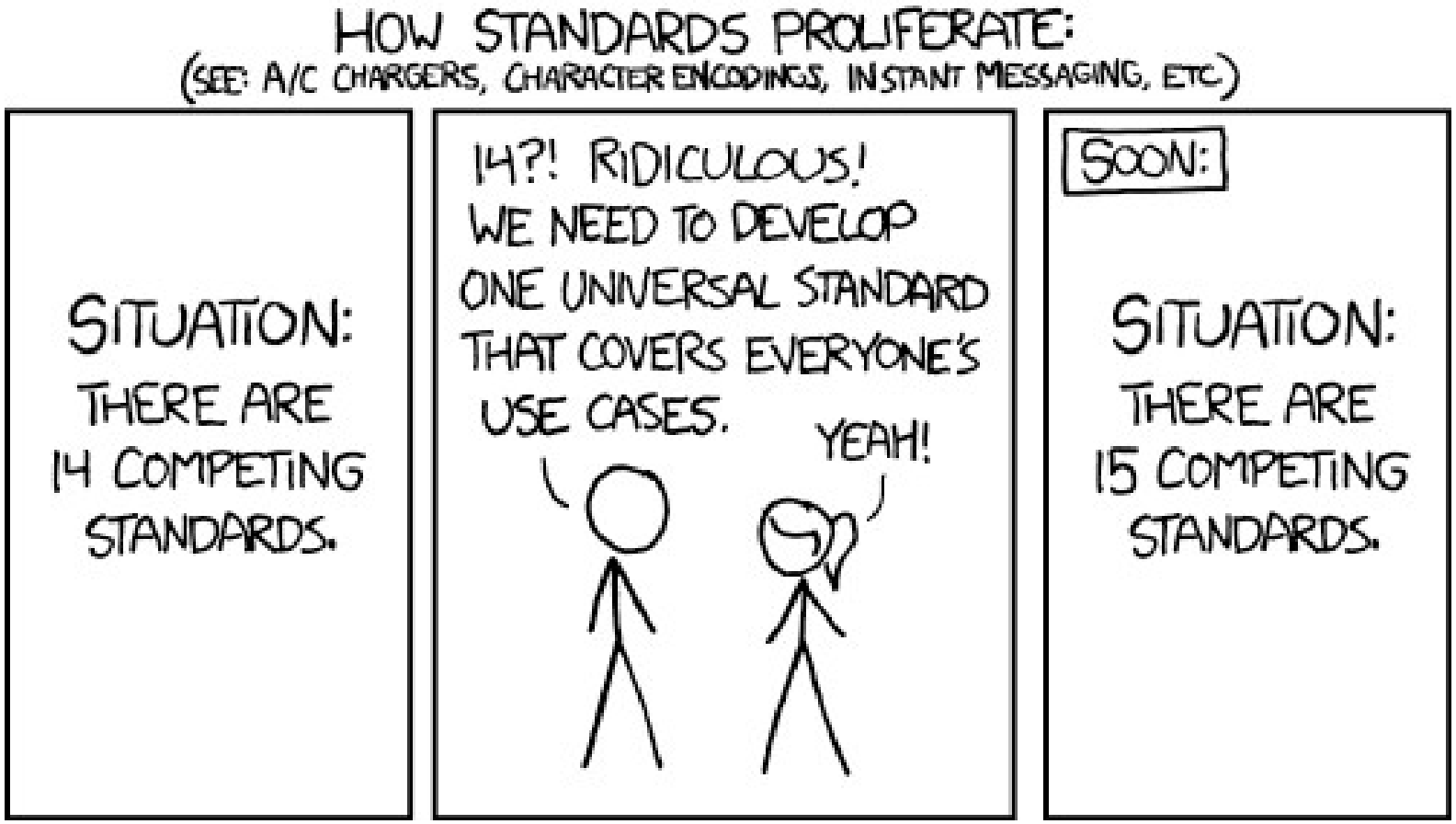
We really need to simplify.

The *real* problem

Everything is too complex.
We really need to simplify.

I've created a tool for this!

- every IT-tool creater ever



https://xkcd.com/927/

The observed trend – according to IT toolmakers

- Developers don’t know sh$t about IT infra and config*
- Developers will never learn IT infra and config
- Developers have no time for IT infra and config because of their PBI’s
- Developers are the customers that need to be served by IT Ops
- Public Cloud (really? Is there something else?)

Tools should aim for a “developer first experience”

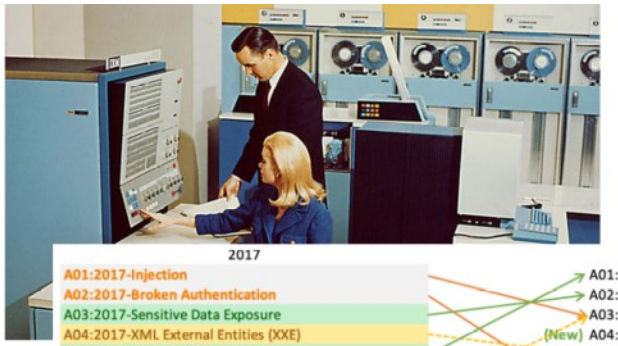
** I agree with this statement up to a certain level (a truly extremely high level)*

My observation(s)

- Developers need to tightly collaborate with IT Ops (hence: DevOps)
  - They don’t, because of <insert reasons here>
- IT Ops struggles to keep their infra and config in control
  - Thet often don’t succeed, because of <insert reasons here>

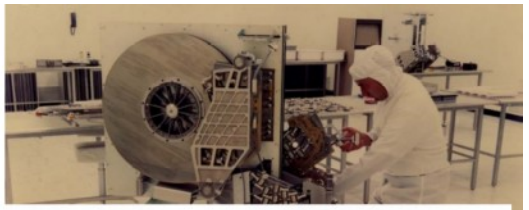
It keeps getting worse...

My observation(s)



splunk>

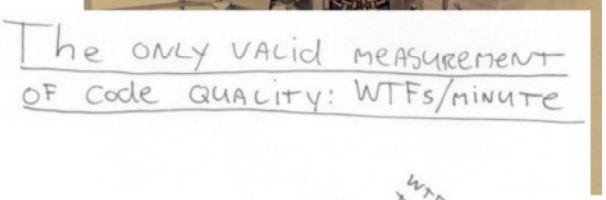
aws

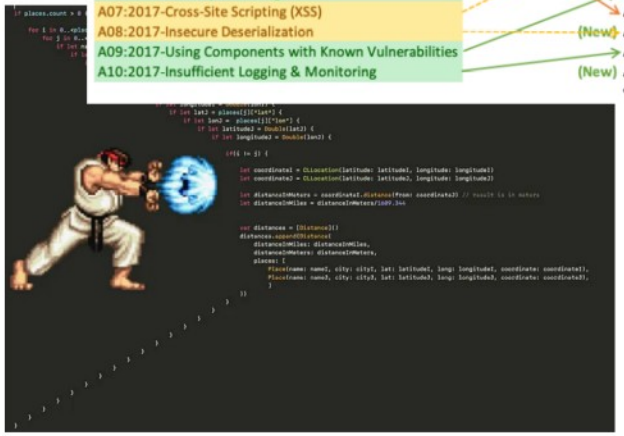


TRM | Red Hat

2017	2021
A01:2017-Injection	A01:2021-Broken Access Control
A02:2017-Broken Authentication	A02:2021-Cryptographic Failures
A03:2017-Sensitive Data Exposure	A03:2021-Injection
A04:2017-XML External Entities (XXE)	(New) A04:2021-Insecure Design
A05:2017-Broken Access Control	A05:2021-Security Misconfiguration
A06:2017-Security Misconfiguration	A06:2021-Vulnerable and Outdated Components
A07:2017-Cross-Site Scripting (XSS)	A07:2021-Identification and Authentication Failures
A08:2017-Insecure Deserialization	(New) A08:2021-Software and Data Integrity Failures
A09:2017-Using Components with Known Vulnerabilities	A09:2021-Security Logging and Monitoring Failures*
A10:2017-Insufficient Logging & Monitoring	(New) A10:2021-Server-Side Request Forgery (SSRF)*

* From the Survey

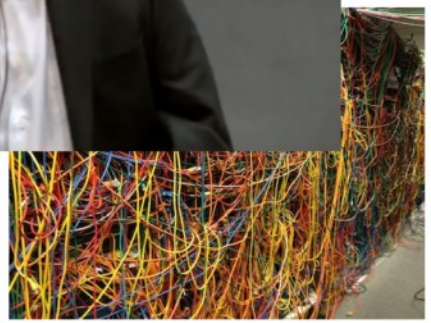




npr

salesforce

mx mendix



Microsoft Windows 2000

SAP

WebSphere. software

https://arstechnica.com/information-technology/2013/09/nvidia-seeks-peace-with-linux-pledges-help-on-open-source-driver/

https://www.backupassist.com/blog/server-cable-disasters-that-look-like-famous-paintings

https://arstechnica.com/information-technology/2014/04/50-years-ago-ibm-created-mainframe-that-helped-bring-men-to-the-moon/

https://en.wikipedia.org/wiki/Pyramid_of_doom_(programming)

https://levelup.gitconnected.com/escape-the-pyramid-of-doom-c58edd326225

https://www.osnews.com/story/19266/wtfsm/

https://commons.wikimedia.org/wiki/Main_Page

https://www.pingdom.com/blog/amazing-facts-and-figures-about-the-evolution-of-hard-disk-drives/

My observation(s)

- We stack layer upon layer upon layer (upon layer)
- We stack tool upon tool upon tool (upon tool)
- We stack library upon library upon library (upon library)
- **NEW:** very high change velocity (who needs LTS-versions anyway these days?)
- Developers who say....

HET WERKT OP MIJN LAPTOP

CaC & IaC 2.0?

How do toolmakers aim to solve this struggle?

<nerd stuff>

Application Programming Interface

<nerd stuff>

Can be used directly through HTTP-requests (GET/PUT/POST) after OAUTH2.0 authentication...

```
# first: create Service Principal (through CLI)
$ az ad sp create-for-rbac \
    --name ${SP_NAME} \
    --display-name ${SP_NAME} \
    --role Contributor --scopes ${SCOPE} \
    --create-cert

# generate JSON Web Token (JWT) based on Service Principal certificate
# Y = Base64URLEncode(header) + '.' + Base64URLEncode(payload)
JWT token = Y + '.' + Base64URLEncode(RSASHA256(Y))

# X5T = Base64url-encoded SHA-1 thumbprint of the X.509 certificate's DER encoding.
# say whut?
$ X5T_STUFF=$(openssl x509 -inform PEM -in "${SP_CERTNAME}" -outform DER | \
    sha1sum | \
    cut -b 1-40 | \
    base64)

# create header
$ TOKEN_HEADER=$(echo -n '{"alg":"RS256","typ":"JWT", x5t:"${X5T_STUFF}"}' | base64 | \
    sed s/\+/-/ | sed -E s/=+$//)

# create payload
$ TOKEN_PAYLOAD=$(echo -n '{"sub":"RS256inOTA","name":"Azure Stuff"}' | \
    base64 | sed s/\+/-/ | sed -E s/=+$//)

# generate the JSON Web Token
$ AZ_JWT=$(echo -n "${TOKEN_HEADER}.${TOKEN_PAYLOAD}" | \
    openssl dgst -sha256 -binary -sign "${SP_CERTNAME}"  | \
    openssl enc -base64 | tr -d '\n=' | tr -- '+/' '-_')
```

https://learn.microsoft.com/en-us/entra/identity-platform/certificate-credentials

https://mauridb.medium.com/calling-azure-rest-api-via-curl-eb10a06127 https://techdocs.akamai.com/iot-token-access-control/docs/generate-jwt-rsa-keys

https://learn.microsoft.com/en-us/cli/azure/azure-cli-sp-tutorial-1?toc=%2Fazure%2Fazure-resource-manager%2Ftoc.json&view=azure-cli-latest&source=post_page-----eb10a06127----------&tabs=bash

<nerd stuff>

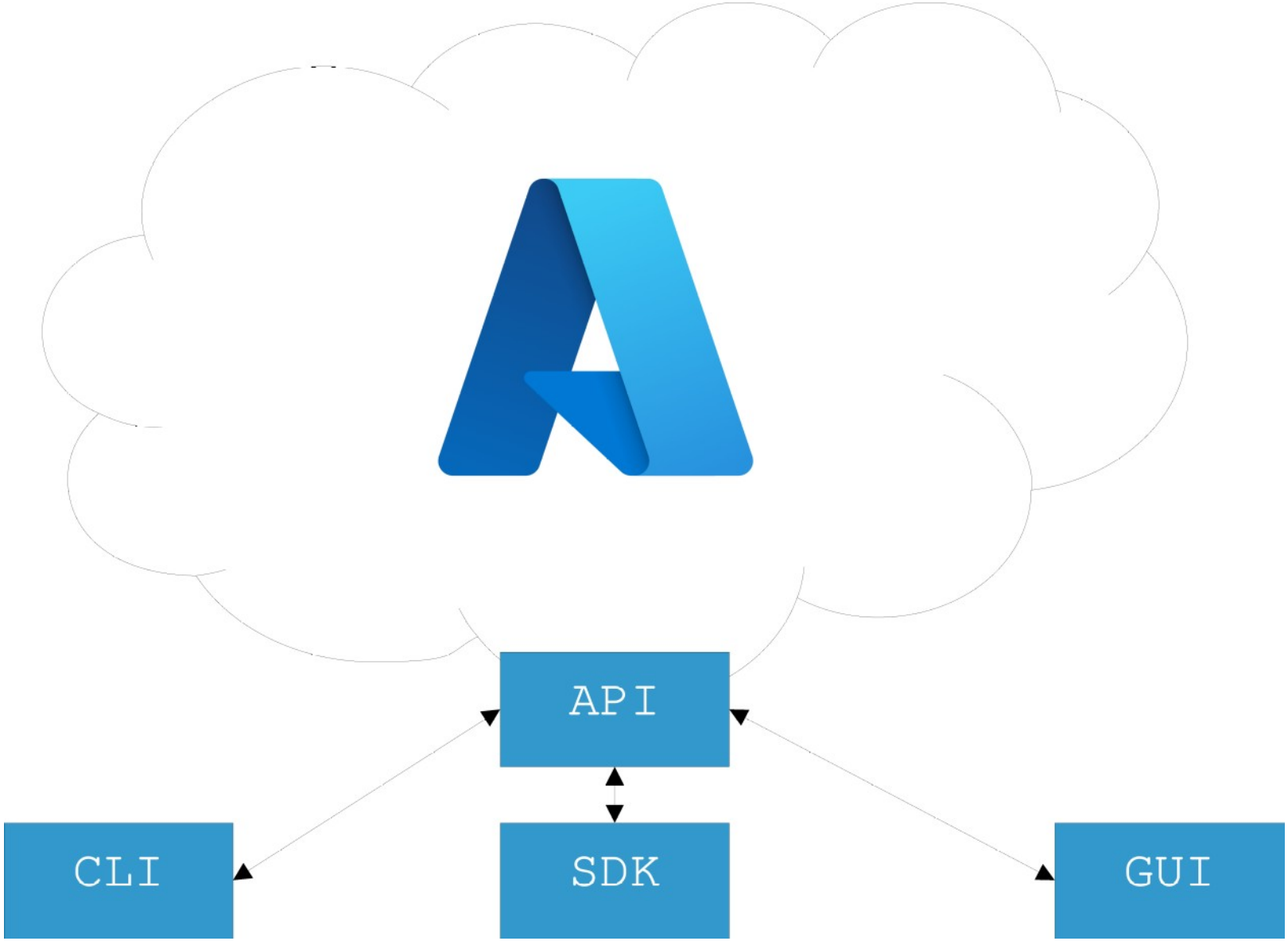
Can be used directly through HTTP-requests (GET/PUT/POST)

```
# use JWT token to request bearer token
$ curl -X POST -d "grant_type=client_credentials&client_id=${SP_APPID}&\
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer&\
client_assertion=${AZ_JWT}&\
resource=https%3A%2F%2Fmanagement.azure.com%2F" \
https://login.microsoftonline.com/${TENANT_ID}/oauth2/token

# request authentication token
$ curl -X GET -H "Authorization: Bearer ${BEARER_TOKEN}" \
-H "Content-Type: application/json" \
https://management.azure.com/subscriptions/${SUBSCRIPTION_ID}\
/providers/Microsoft.Web/sites?api-version=2016-08-01
```

https://mauridb.medium.com/calling-azure-rest-api-via-curl-eb10a06127

https://learn.microsoft.com/en-us/entra/identity-platform/v2-oauth2-client-creds-grant-flow?source=post_page-----eb10a06127----------#second-case-access-token-request-with-a-certificate

<nerd stuff>



<nerd stuff>

Can be used via abstraction: CLI & SDK
(GUI != 4 n3rdz)

```
# authentication with CLI. $SP_CERTNAME = path to cert
$ az login -username ${SP_APPID} --password ${SP_CERTNAME}
```

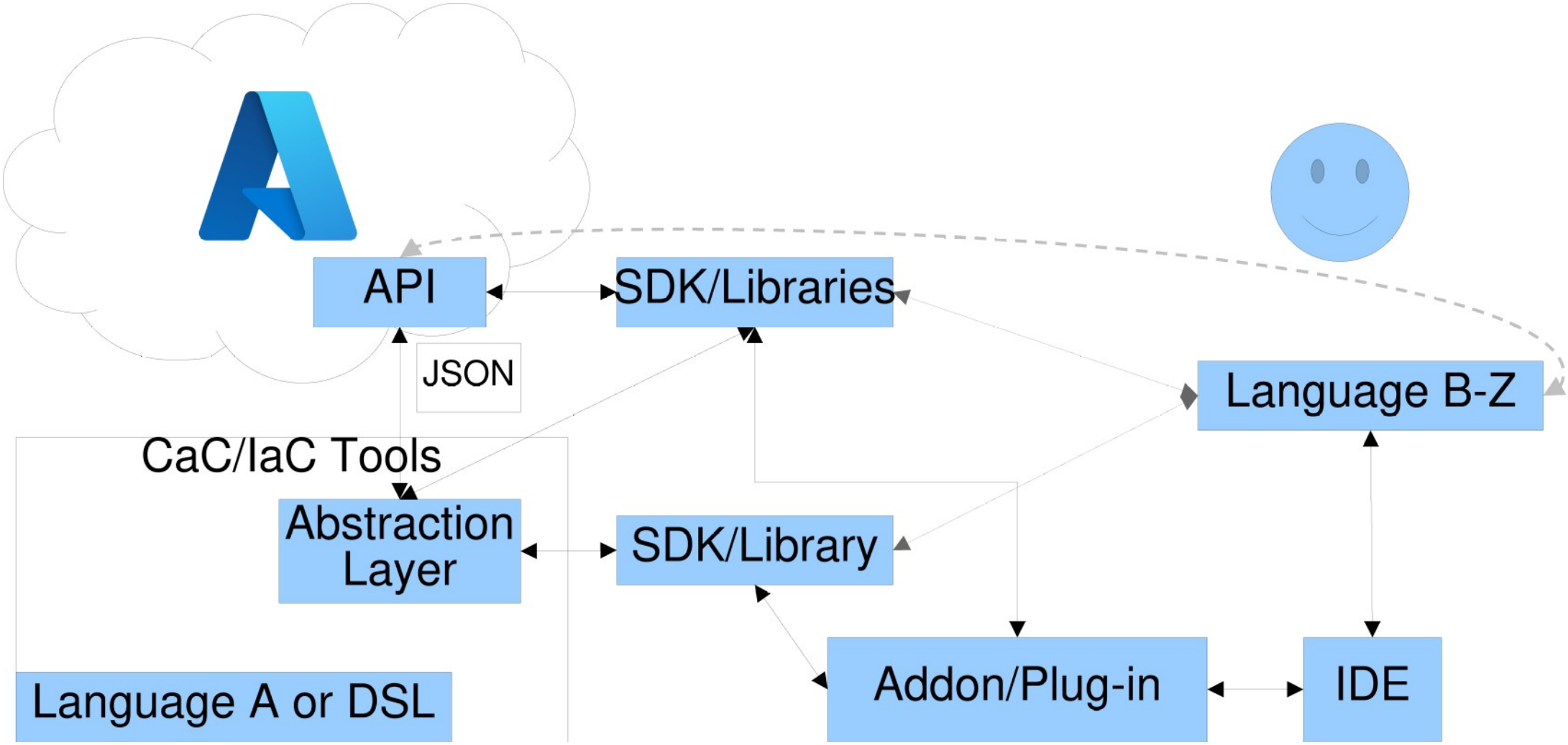
```
# authentication with Python + SDK
```

```python
import os

from azure.identity import DefaultAzureCredential
from azure.mgmt.compute import ComputeManagementClient
from azure.mgmt.network import NetworkManagementClient
from azure.mgmt.resource import ResourceManagementClient

credential = DefaultAzureCredential()
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]
resource_client = ResourceManagementClient(credential, subscription_id)
```

https://learn.microsoft.com/en-us/cli/azure/azure-cli-sp-tutorial-3

https://learn.microsoft.com/en-us/azure/developer/python/sdk/examples/azure-sdk-example-virtual-machines?tabs=bash

<nerd stuff>


API
SDK/Libraries
JSON
Language B-Z
CaC/IaC Tools
Abstraction Layer
SDK/Library
Addon/Plug-in
IDE
Language A or DSL

</nerd stuff>

What’s happening in Config & Infrastructure Land?

WingLang

- What it is: A programming language for the cloud
- Current version: ~v0.59.xx
- Written in: TypeScript
- Website: https://winglang.io
- Github: https://github.com/winglang/wing/releases/latest
- Maturity: toddler
- Maintained by: Wing Cloud (Israel)
- License(s): partly MIT, partly mixture, partly Wing Cloud specific
- Backing: $20M

winglang

WingLang

main.w

```
bring cloud;
bring util;
bring expect;

let bucket = new cloud.Bucket();
let queue = new cloud.Queue();

queue.setConsumer(inflight (message) => {
  bucket.put("wing.txt", "Hello, {message}");
}, timeout: 30s);

test "Hello, world!" {
  queue.push("world!");

  let found = util.waitUntil(() => {
    log("Checking if wing.txt exists");
    return bucket.exists("wing.txt");
  });

  expect.equal(bucket.get("wing.txt"), "Hello, world!");
}
```

wing compile --platform tf-aws main.w

main.tfaws

terraform apply --yolo

main.tfazure

AWS S3 bucket

Azure Blob Storage

wing compile --platform tf-azure main.w

winglang

Crossplane

- What it is: The cloud native control plane framework
- Current version: ~v1.15
- Written in: Go
- Runs as: deployment on K8s
- Website: https://crossplane.io
- Github: https://github.com/crossplane/crossplane/releases/latest
- Maturity: teenager
- Maintained by: The Linux Foundation / CNCF (almost graduated)
- License(s): Apache 2.0

Crossplane

Everything is a K8s deployment...
Custom Resource Definition (CRDs) = abstraction layer

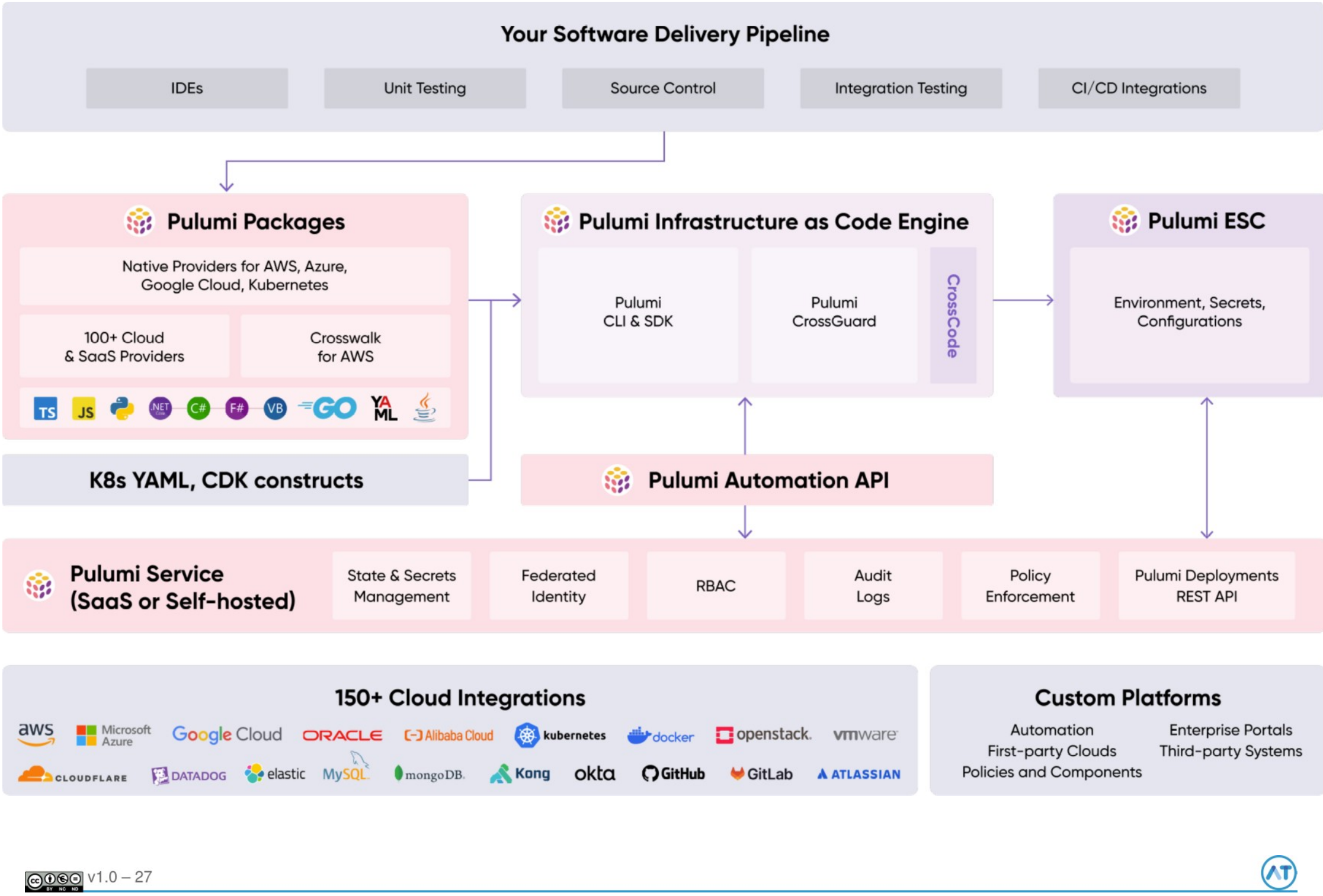
```
cat <<EOF | kubectl apply -f -
apiVersion: pkg.crossplane.io/v1
kind: Provider
metadata:
  name: provider-azure-network
spec:
  package: xpkg.upbound.io/upbound/provider-azure-network:v0.34.0
EOF
```

```
cat <<EOF | kubectl create -f -
apiVersion: network.azure.upbound.io/v1beta1
kind: VirtualNetwork
metadata:
  name: crossplane-quickstart-network
spec:
  forProvider:
    addressSpace:
      - 10.0.0.0/16
    location: "Sweden Central"
    resourceGroupName: docs
EOF
```

Pulumi

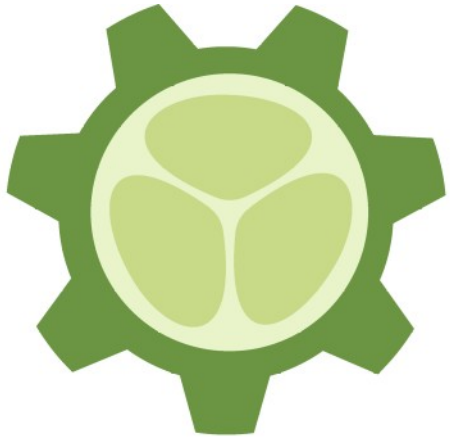
- What it is: Infrastructure as Code in any Programming Language
- Current version: ~v3.108.x
- Written in: Go
- Website: https://crossplane.io
- Github: https://github.com/pulumi/pulumi/releases/latest
- Maturity: young adult
- Maintained by: Pulumi Corporation (US privately held company)
- License(s): Apache 2.0
- Backing: $98.5M

Pulumi


Your Software Delivery Pipeline
IDEs
Unit Testing
Source Control
Integration Testing
CI/CD Integrations
Pulumi Packages
Native Providers for AWS, Azure, Google Cloud, Kubernetes
100+ Cloud & SaaS Providers
Crosswalk for AWS
TS
JS
.NET
C#
F#
VB
GO
YAML
K8s YAML, CDK constructs
Pulumi Infrastructure as Code Engine
Pulumi CLI & SDK
Pulumi CrossGuard
CrossCode
Pulumi Automation API
Pulumi ESC
Environment, Secrets, Configurations
Pulumi Service (SaaS or Self-hosted)
State & Secrets Management
Federated Identity
RBAC
Audit Logs
Policy Enforcement
Pulumi Deployments REST API
150+ Cloud Integrations
aws
Microsoft Azure
Google Cloud
ORACLE
Alibaba Cloud
kubernetes
docker
openstack.
vmware
CLOUDFLARE
DATADOG
elastic
MySQL
mongoDB.
Kong
okta
GitHub
GitLab
ATLASSIAN
Custom Platforms
Automation
First-party Clouds
Policies and Components
Enterprise Portals
Third-party Systems

Pkl

- What it is: an embeddable configuration language
- Current version: ~v0.25.x
- Written in: mostly Java & Kotlin
- Website: https://pkl-lang.org/
- Github: https://github.com/apple/pkl/releases/latest
- Maturity: toddler (new to the public) / adult (used within Apple for >10 years)
- Maintained by: Apple Inc (US public company)
- License(s): Apache 2.0
- Backing: Apple (worth roughly ~$2.7T (~2.7x Dutch economy)



Pkl

```yaml
apiVersion: v1
kind: Service
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  # comment or delete the following line if you want to use a LoadBalancer
  type: NodePort
  # if your cluster supports it, uncomment the following to automatically create
  # an external load-balanced IP for the frontend service.
  # type: LoadBalancer
  ports:
    - port: 80
  selector:
    app: guestbook
    tier: frontend
```

Kubernetes'

OpenAPI spec

deployment.yaml

```
,,---
import "@k8s/K8sResource.pkl"
import "@k8s/api/apps/v1/Deployment.pkl"
import "@k8s/api/core/v1/Service.pkl"
```

```
resources: Listing<K8sResource> = new {
  new Service {
    metadata {
      name = "frontend"
      labels {
        ["app"] = "guestbook"
        ["tier"] = "frontend"
      }
    }
    spec {
      type = "NodePort"
      ports {
        new {
          port = 80
        }
      }
      selector {
        ["app"] = "guestbook"
        ["tier"] = "frontend"
      }
    }
  }
}
```

deployment.pkl

For the record....

- Mgmt Config *(James Shubin)*
  - Nice guy, fun tool, not finished
  - https://purpleidea.com/projects/mgmt-config/
- System Initiative *(Adam Jacob (founder of Chef))*
  - Great speaker, tool looks quite promising, but not finished
  - https://www.systeminit.com/

#rant

- How many people are able to still understand what the h@ck is going on?
- How will an average IT department deal with all this stuff?
- Do we really need it?
  - What real world problem(s) are we actually solving?
  - You still need to learn a lot of new stuff anyway...
    - A new config or programming language
    - New tools
    - New abstraction principles
    - Fundamentals of IT infrastructure stay put (things like CIDR blocks....)

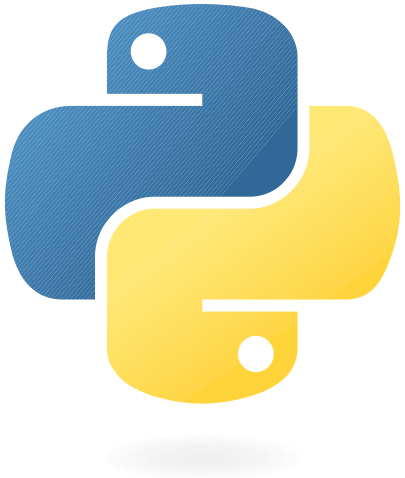
Verdict – option #1

Knock yourself out with tools

&

enjoy keeping all the balls in the air

Verdict – option #2



python™