



High Throughput Molecular Pathology: PathOS & Docker & Anomaly

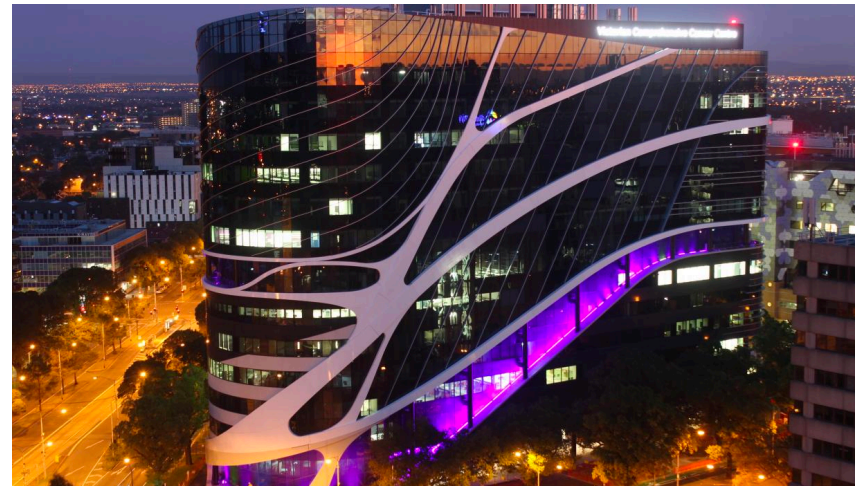
Dr Kenneth Doig, Dr Thomas Conway
October 2019

<https://github.com/PapenfussLab/PathOS>

PathOS: Web Based Variant Curation

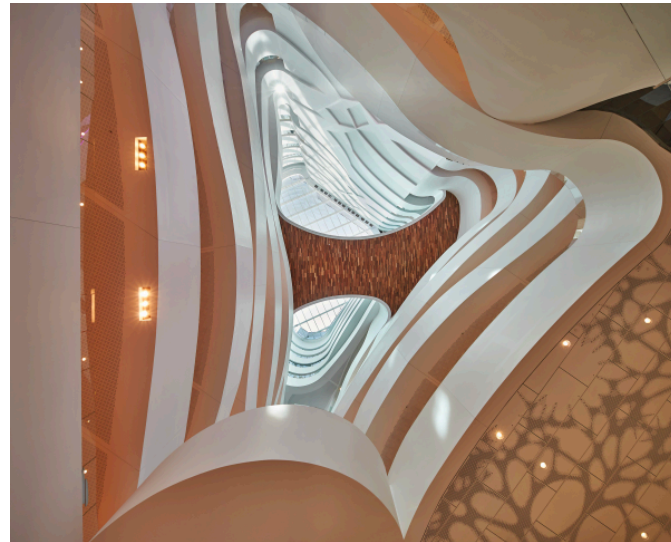
The Molecular Pathology Department at Peter MacCallum Cancer Centre (PMCC) reports on over 1,000 patient samples per month with HTS based assays.

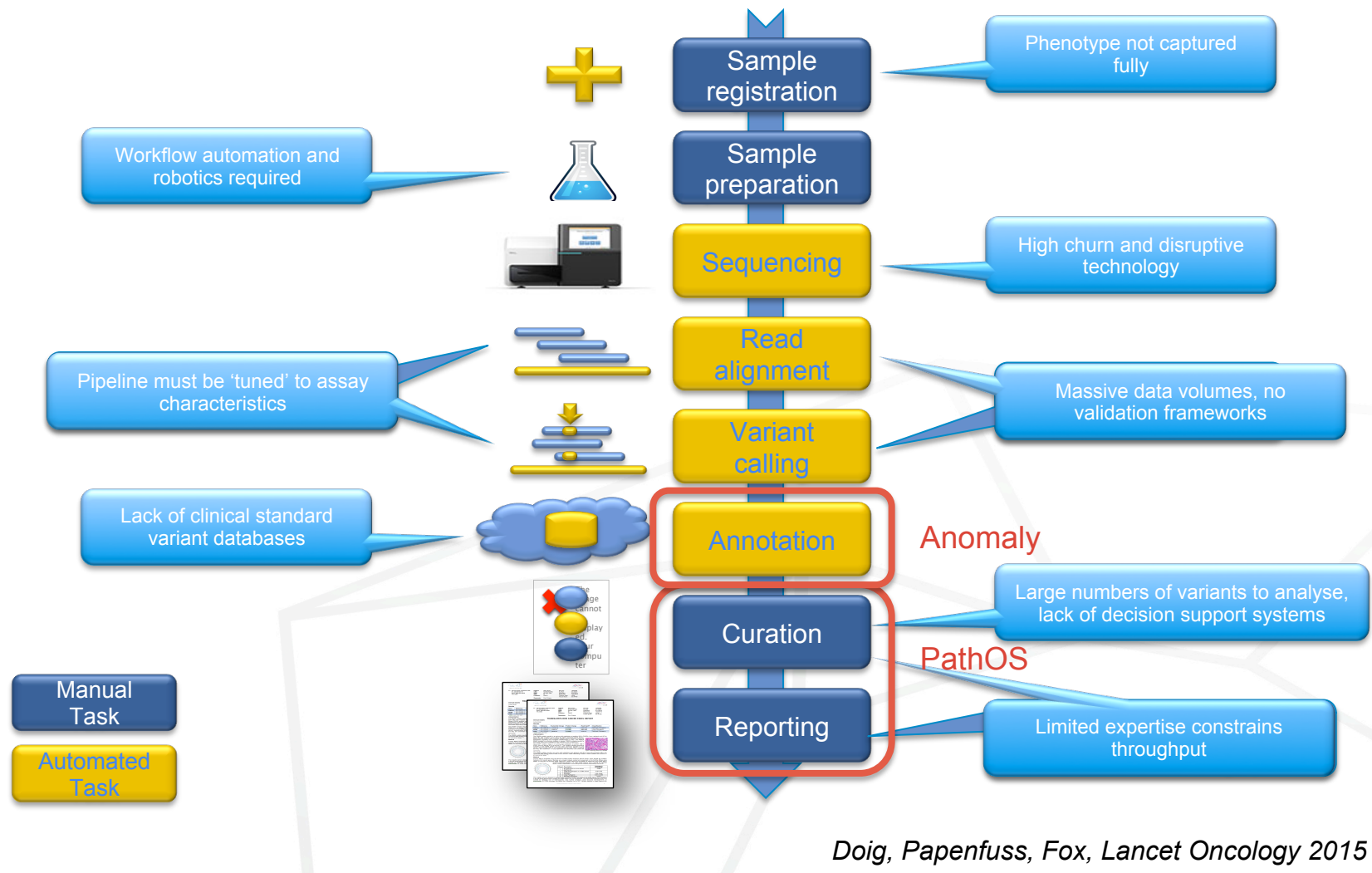
PathOS is the in-house variant curation platform, which spans some variant prioritisation, curation, and reporting.



PathOS: Web Based Variant Curation

- Small amplicon panels with a few genes
 - ~80% of samples;
 - 4-50 genes;
 - 1,000-2,000 X coverage
- Larger hybrid-capture panels with a few hundred genes
 - ~20% of samples;
 - ~50-500 genes;
 - 100-500 X coverage





PathOS: Web Based Variant Curation

PathOS has several components:

- Tomcat/Grails web application
- MariaDB (MySQL) back-end database
- Httpd for serving VCF/BAM for IGV
- Grails/Hibernate data ingestion
 - YAML/JSON for patient data, sequencing metadata
 - VCF/BAM for variants and data viewing
- Aspose templating for reports
- Annotation servers for enriching data
- HL7 middleware for patient and assay data (translates to YAML/JSON).

External components used by the clinical scientist include:

- Alamut
- Hospital LIMS / EMR
- IGV
- Google Scholar
- Other assays (Sanger,PCR etc)
- Spreadsheets !

Doig et. al. Genome Med. 2017 Apr 24;9(1):38

PathOS: Web Based Variant Curation

- Current deployment has multiple daemons on (virtual) Linux hosts.
 - MariaDB, Tomcat, Metadata loader, Variant loader, Babble (HL7 middleware), Httpd
- For various reasons specific versions of some components are required.
- The evolution of the system is gradually separating site-specific code and configuration from core code and configuration.
 - Patient, Assay, Sequencing metadata are YAML/JSON
 - As much isolation of dependency on specific locations for things
 - Allowing multiple instances to co-exist on a single server
(Production, UAT, Development, 3xResearch, Training (x lots!), Cloud)

Virtual Machines vs Containers

We initially set up a VirtualBox VM with all the components for public access.

- Configuration requires logging in and editing files.
- Mounting filesystems to serve VCF/BAM files for IGV is complicated.
- VMs are bulky, and there is not a standard description for how to configure it when the software is updated.

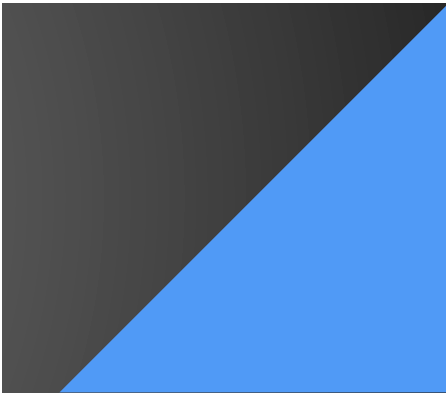
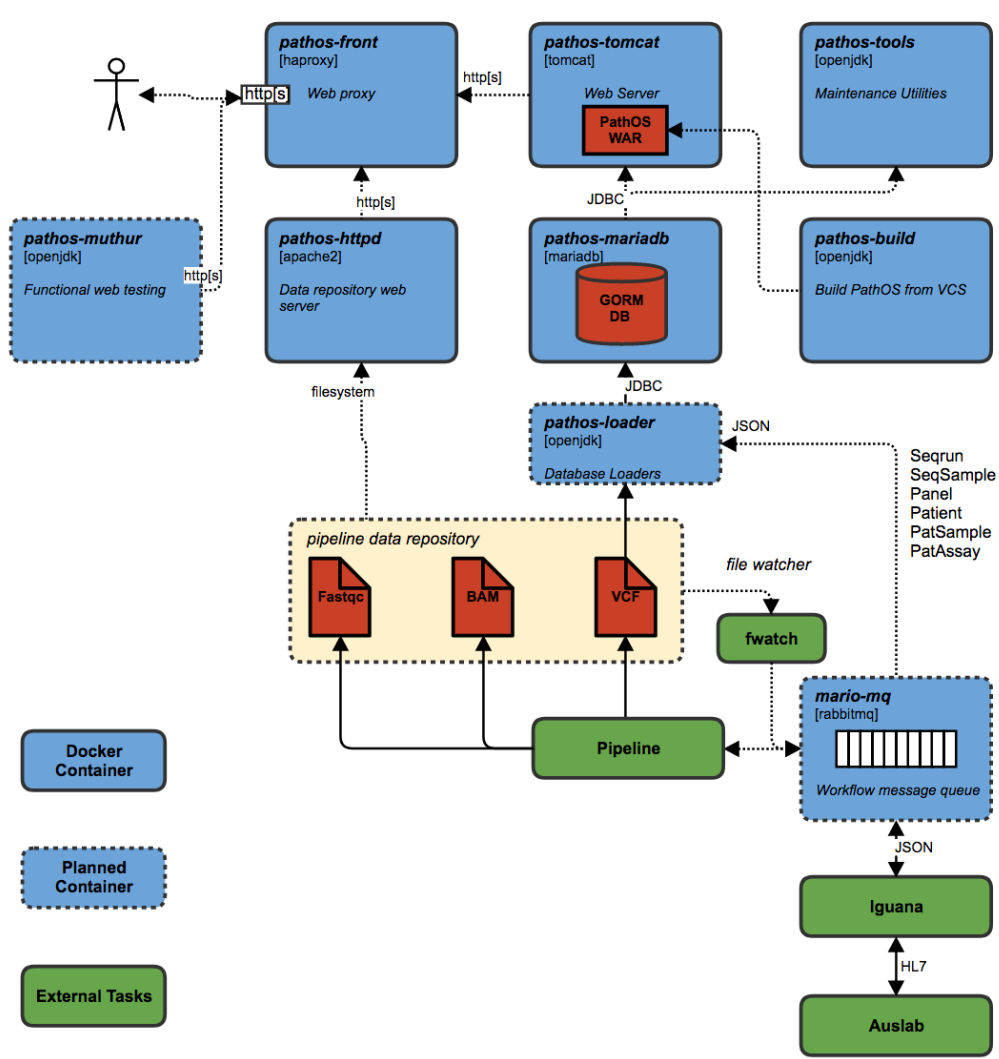
Switching to Docker simplifies things greatly.

- Even the initial complicated dockerization was simpler because all the steps were captured:
 - Dockerfile for each separate component
 - Docker-compose to link them up
- Private network simplifies naming, credentials, & security
- The overlay filesystem makes it simple to add or replace files.

First Docker Version

Initially we had a docker-compose that made use of custom images for each service:

- Database with pre-configured tables
- Tomcat with WAR file, and configuration
- JDK image for running sundry command-line tools
- Httpd for serving VCF/BAM files
- Haproxy to put everything behind a single HTTP/HTTPS port



To Image or not to Image?

Putting things in a custom image:

- Can be very bulky
- Can reduce flexibility
- Makes updates harder
- Makes docker-compose simple

Using a generic image:

- Most updates won't require repo access
- Requires packaging of auxiliary data
 - Config files, default tables, scripts, ...
- Requires more complicated docker-compose files.
- Shared credentials are obvious
 - We should use **secrets**

Where we have got to

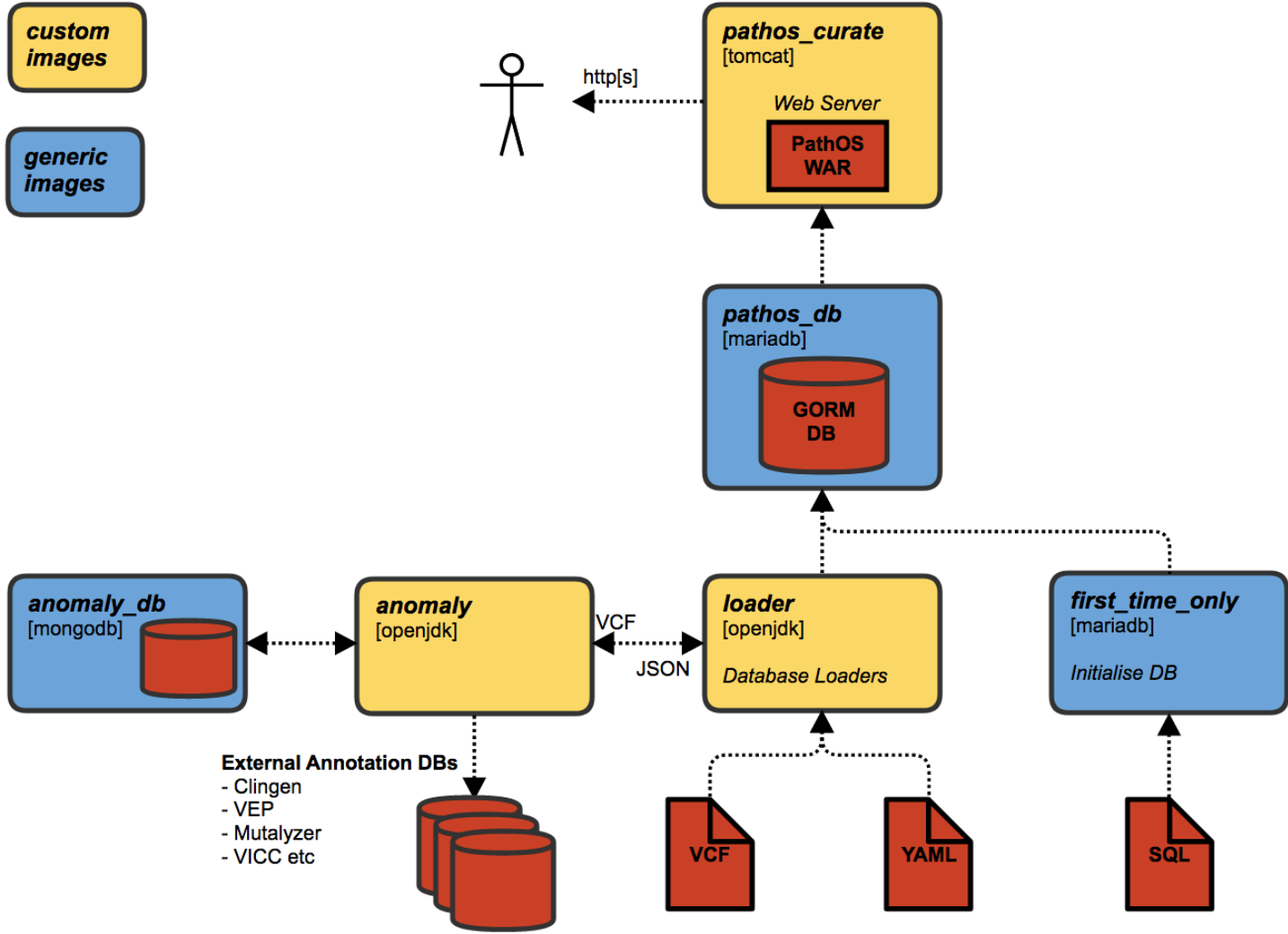
As things have evolved, we've arrived at a much simpler setup.

- **Generic Database**
 - Use docker-compose volumes for persistence across restarts
 - Use a “first-time-only” service to run some setup scripts (is there a better way?)
- **Custom Tomcat image with WAR file, and basic configuration**
 - Serve BAM/VCF from the tomcat itself, so no Httpd required (avoids CORS problems)
 - Use volume to overwrite the configuration where necessary
- **Custom JDK image to load data**
 - `docker run -v <path-to-input-data>:/input-data.d/:ro loader`
 - Will load YAML/JSON metadata, or VCFs
 - Pipes through `gunzip -c` if necessary

Current Setup

custom images

generic images



Where we have got to

Startup order is an issue.

- Required behaviour:
 - Mariadb starts and becomes ready
 - Grails app (on Tomcat) starts and creates tables in database (with some problems)
 - “First-time-only” scripts run to fix the tables and install some required rows in some tables
- Docker-compose has different behaviour in different versions
 - V2 allows “depends_on” and healthchecks
 - V3 has weaker constraints to facilitate swarm
- We use V2, with healthcheck configs to enforce dependencies

Where we have got to

It's not perfect yet

- It would be good if the configuration could be overwritten with environment variable
 - Like MYSQL_DATABASE, MYSQL_USER, MYSQL_PASSWORD, etc.
- It would be really nice if there was builtin support for mounting zip-files in compose
 - `curate:`

```
image: tomcat7:jre7
depends_on:
  - pathosdb
volumes:
  - ${PWD}/config-bundle.zip:/etc/pathos/:ro
```

Containers for testing

Creating regression tests for a complex system is non-trivial:

- Putting the system in a precisely known state
- Testing error recovery can require “damaging” things
- Cleaning up after so you don’t leave detritus around

Docker[-compose] is awesome for these things:

- Simple “run this” tests can use a one-shot invocation
- More complex tests start up in daemon mode

A simple test: transform some data

Wrapper shell script

```
#!/bin/bash
set -e
mkdir output
docker-compose -f 006-docker-compose.yaml up
docker-compose -f 006-docker-compose.yaml down
./yamldiff 006-input.yaml output/output.yaml
rm -rf output
```

docker-compose.yaml

```
version: '2.1'
services:
  babble:
    image: openjdk:7-jre
    volumes:
      - ./babble-server.jar:/babble/lib/babble-server.jar:ro
      - ./006-babble-config.yaml:/babble/babble-config.yaml:ro
      - ./006-input.yaml:/babble/input.yaml:ro
      - ./output:/babble/output/:rw
    working_dir: '/babble'
    command: ['java', '-jar', 'lib/babble-server.jar',
              '-f', 'babble-config.yaml']
```


A simple test: database test

Wrapper shell script

```
#!/bin/bash
set -e
rm -rf output 2>/dev/null
mkdir output
docker-compose -f 012-docker-compose.yaml up -d
sleep 15
docker exec -i tests_test-database_1 \
  mysql -uroot -px \
    < 012-query.sql > output/output.txt
docker-compose -f 012-docker-compose.yaml down
diff -u 012-output.txt output/output.txt
rm -rf output
```

docker-compose.yaml

```
version: '2.1'
services:
  babble:
    [almost same as before]
  test-database:
    image: mariadb
    environment:
      MYSQL_ROOT_PASSWORD: x
    volumes:
      - ./012-initdb.sql:/docker-entrypoint-initdb.d/00-initdb.sql:ro
    healthcheck:
      test: ['CMD', 'mysqladmin', '-uroot', '-px', 'version']
      interval: 10s
      timeout: 10s
      retries: 10
```

Singularity

Some relevant differences between Singularity and Docker

- Docker containers run as root
- Docker is oriented towards micro-services
- Singularity is oriented toward HPC workflows
- Docker makes Hospital IT departments nervous

Network isolation is important for us simplifying our configuration, since everything exists on a private network.

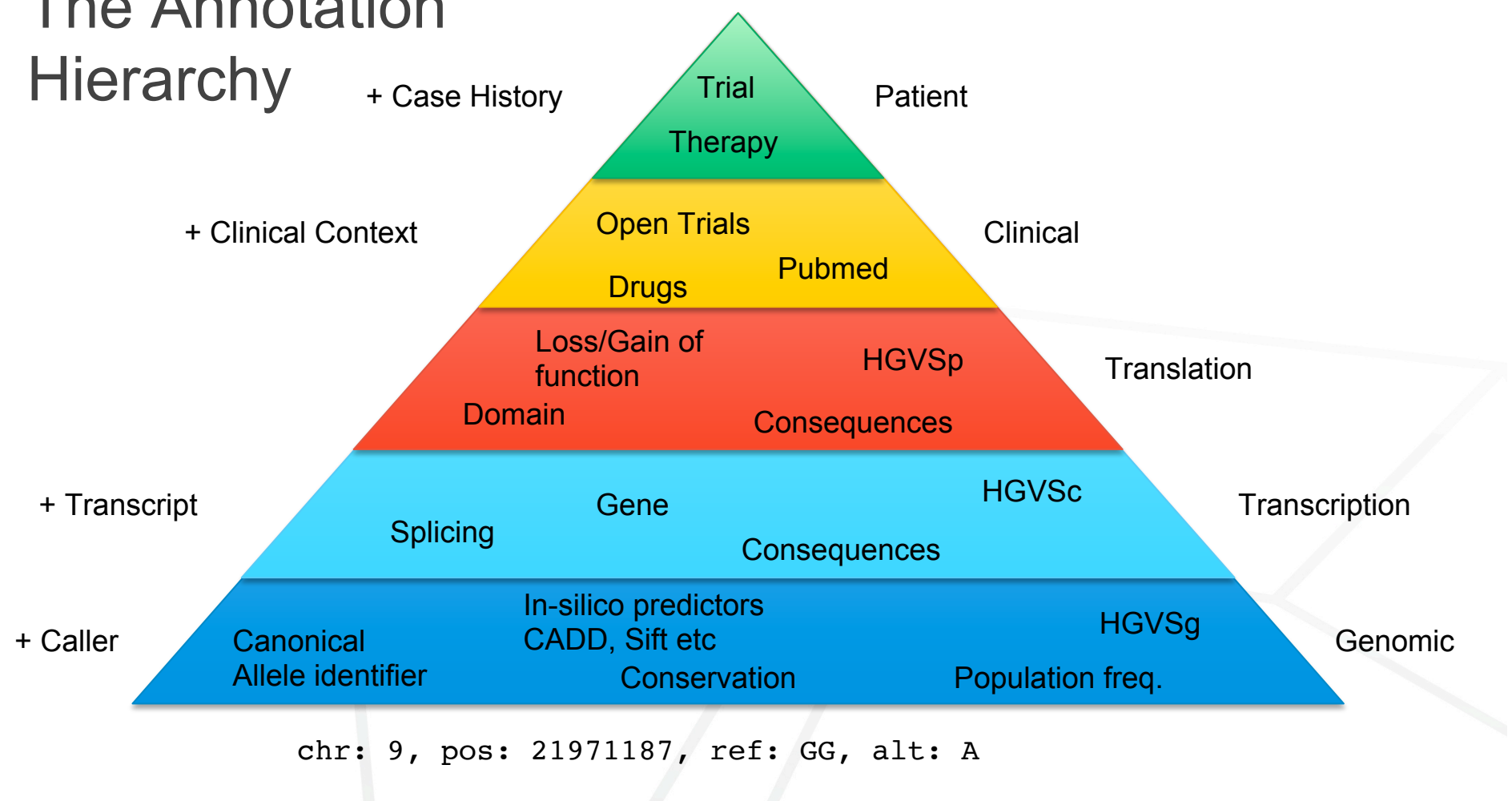
Annotation as a Service (AaaS)

Annotation:

“The process of enriching pipeline variants with genomic, biological and clinical data to inform decision making”

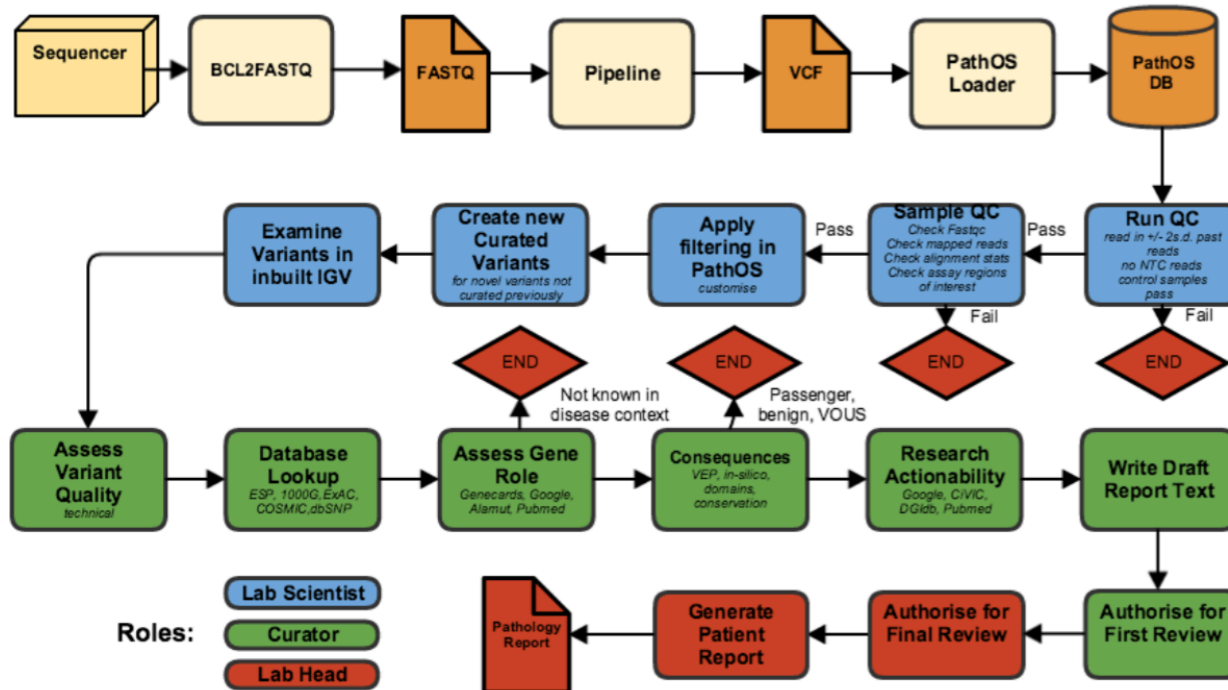
- Anomaly is a web service for annotating variants with a rich set of attributes from public DBs and clinical APIs.
- Microservice not a database
- Created for cloud deployments to save installing 100's of Gb of Annotation Data from many sources
- Get best available annotations at the time of curation
- Provide a schema to configure data sources (URLs) and attributes (JSONPath)
- Can create a synthetic data source
- Cache for retrieval efficiency

The Annotation Hierarchy














Thanks, questions ?

PathOS Workflow



Current Sources

Identifier	Description	Type
VCF	All unpacked attributes from input VCF file	
Mutalyzer	SNV description standardisation	
ClinGenAR	ClinGen Allele Registry	
VEP	Variant effect predictor (consequences)	  
MyVariant	Variant data aggregation service	  
ClinGenER	ClinGen Evidence Registry	
VICC	Variant Interpretation for Cancer Consortium	 X 6

Easily extended for any variant REST API

 Genomic
  Transcript
  Translation
  Clinical

Create your own data source

