

# ARC LOCAL submission plugin

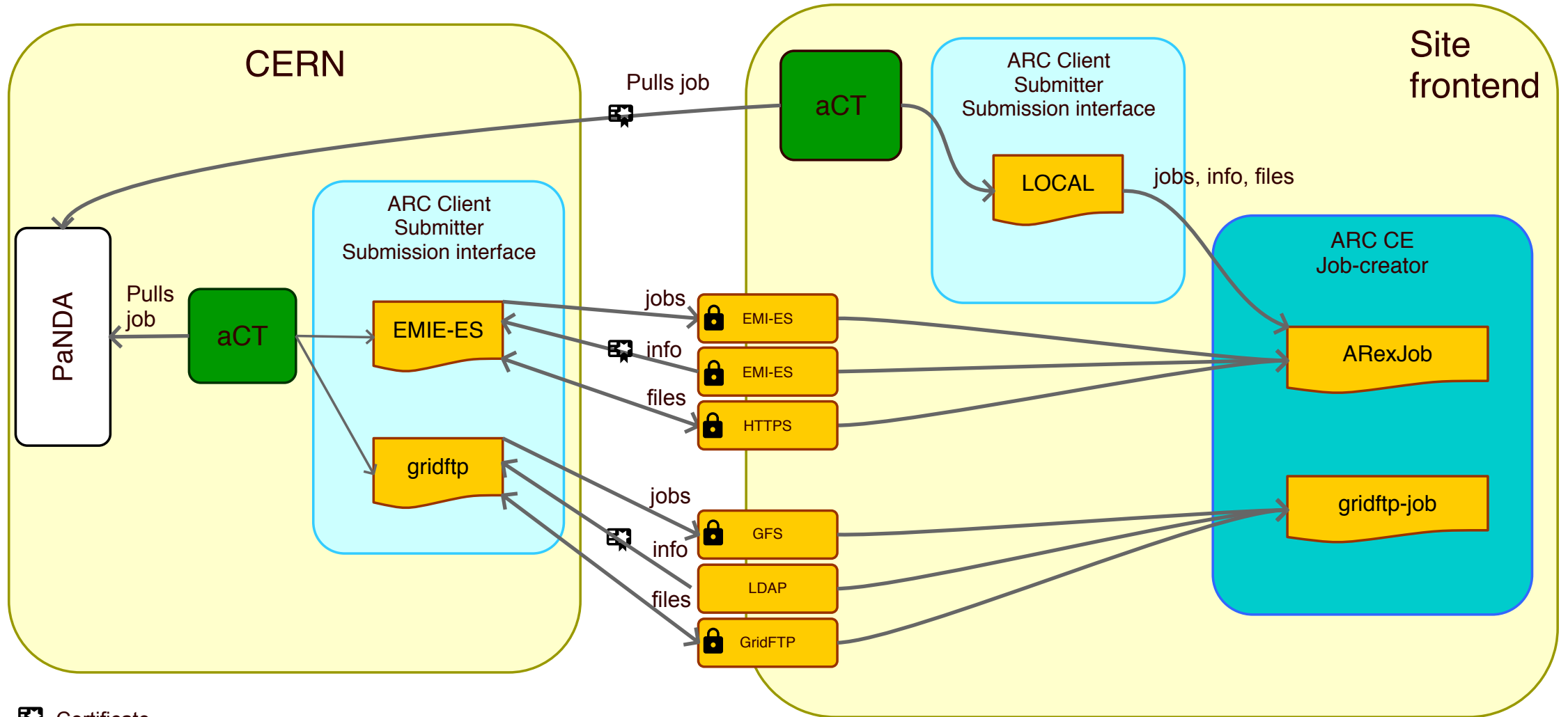
Implementation with aCT on  
an on-demand openstack cluster

ATLAS qualification task

Andrej Filipcic

David Cameron

# The broad picture



- Certificate
- Credentials

# Elastycluster used to set up proof-of-principle implementation of LOCAL plugin

## Why elastycluster?

- Asked to set up a proof-of-principle grid cluster on the University cloud (UH-iaaS).
- Elastycluster used for this in Bern
- On-demand cluster on cloud provider setup with all necessary services and configuration
  - SLURM
  - NFS
  - ...
- Decided to use the Elastycluster also to set up proof-of-principle cluster for the aCT + ARC CE LOCAL submission plugin

# Elasticcluster

<http://elasticcluster.readthedocs.io/en/latest/>

A collection of ansible scripts to set up a cluster on a cloud service

- Ansible scripts are yaml, organized in so-called plays or playbooks, with roles, tasks, templates (++)
  - Roles can be e.g. frontend or compute node, slurm master and so on
  - Tasks can be e.g. install arc, reboot cluster etc
  - Templates: e.g. arc.conf template
  - Plays: instructions of what machines should be run with what tasks
- Supported cloud providers
  - ec2\_boto
  - Google
  - **Openstack**
  - libcloud

anaconda	easybuild	glusterfs-server	hadoop.yml	htcondor.yml	jupyterhub	lua	pbs+maui	r.yml	spark-master
ansible	ganglia-gmetad	glusterfs.yml	hdfs-datanode	iptables	jupyterhub.yml	mcr	pbs+maui.yml	slurm-client	spark-worker
ansible.yml	ganglia-gmond	gridengine-common	hdfs-namenode	ipython	kubernetes-common	mcr.yml	pdsh	slurm-common	yarn-master
bigtop	ganglia-web	gridengine-exec	hive	ipython.yml	kubernetes-master	nfs-client	postgresql	slurm-master	yarn-worker
ceph	ganglia.yml	gridengine-master	hive-server	jenkins	kubernetes-worker	nfs-server	pvfs2	slurm-worker	
ceph.yml	glusterfs-client	gridengine.yml	hpc-common	jenkins.yml	kubernetes.yml	nis	pvfs2.yml	slurm.yml	
common	glusterfs-common	hadoop-common	htcondor	jupyter	lmod	ntp	r	spark-common	

## Playbooks distributed with elasticcluster

Ansible

**SLURM**

GridEngine

HTCondor

**Ganglia**

IPython cluster

Hadoop + Spark

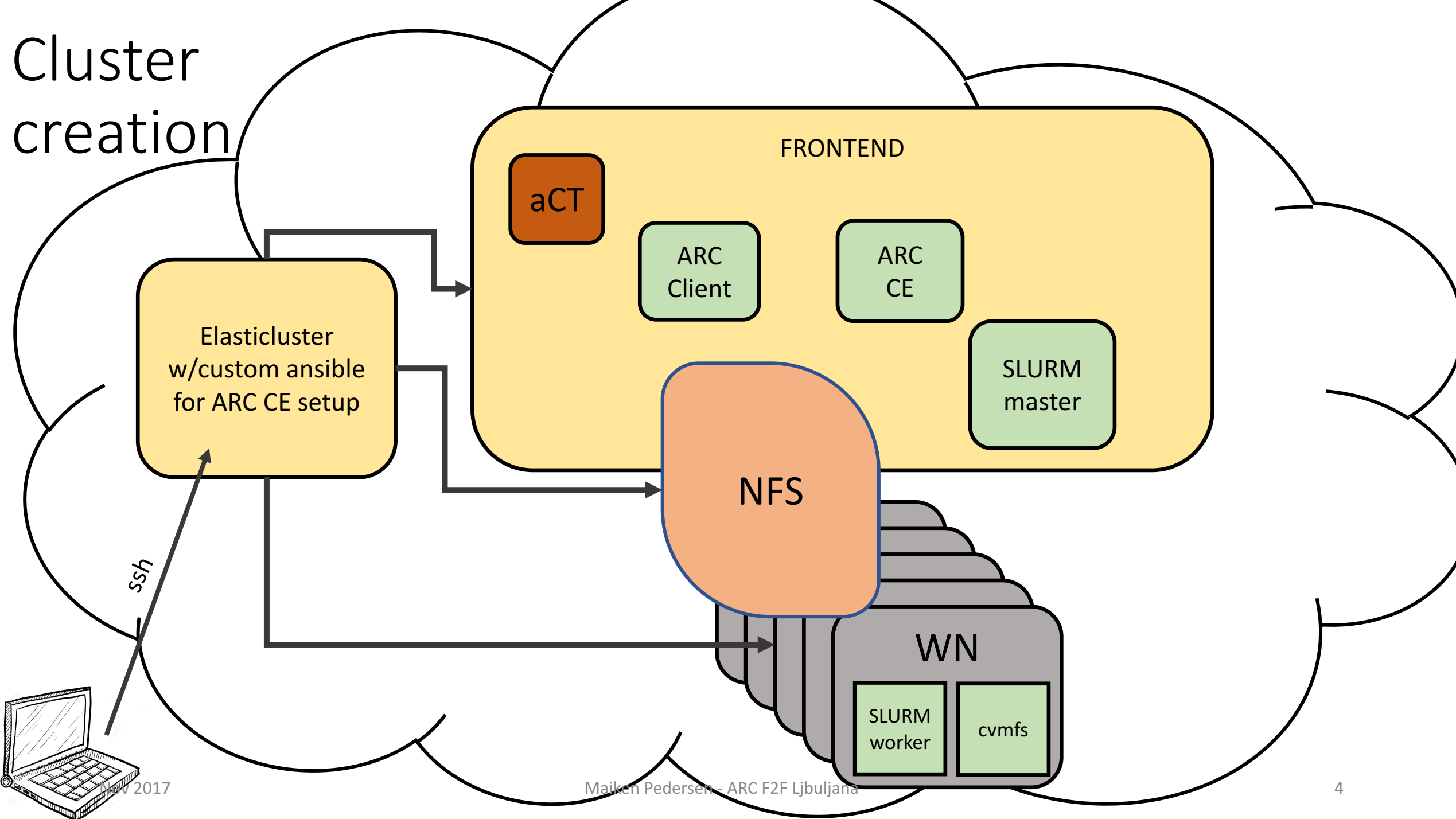
CephFS

GlusterFS

OrangeFS/PVFS2

Kubernetes

# Cluster creation



# Example configuration of elasticcluster

- Elasticcluster contacts the cloudprovider
- Fires up specified number of frontends and compute nodes with specified os and size
- Security group already set up with ssh ports on the UH IaaS dashboard
  - Which ports to allow open for ssh, https etc
- Installs slurm server and client
- Sets up NFS
- Sets up monitoring through ganglia
  - However, not used it yet
- Specific setups with own ansible scripts:
  - ARC + cvmfs
  - aCT

```
[cloud/iaas]
provider=openstack
auth_url=https://api.uh-iaas.no:5000/v3
username=maiken.pedersen@usit.uio.no
password=xxxxxx
project_name=uiu-test-hpc-grid
user_domain_name=dataporten
project_domain_name=dataporten
region_name=osl
identity_api_version=3
```

```
[login/centos]
image_user=centos
image_user_sudo=root
image_sudo=True
user_key_name=cloud
user_key_private=~/.ssh/cloud.key
user_key_public=~/.ssh/cloud.key.pub
```

```
[setup/ansible-slurm]
provider=ansible
frontend_groups=slurm_master,ganglia_master,ganglia_monitor,frontend,cluster
compute_groups=slurm_worker,ganglia_monitor,compute,cluster
global_var_multiuser_cluster=no
```

```
[cluster/slurm]
cloud=iaas
login=centos
setup=ansible-slurm
security_group=default
image_id=df3dedc6-f98c-4eb0-b77e-7f8f24f857e4
frontend_nodes=1
compute_nodes=1
ssh_to=frontend
network_ids=c97fa886-592e-4ad1-a995-6d55651bed78
```

```
[cluster/slurm/frontend]
flavor=m1.medium
```

```
[cluster/slurm/compute]
```

# Creating an ARC-CE with aCT and preparing compute nodes

On frontend:

- ARC, aCT
- Install, configure, and start both

On compute node(s)

- cvmfs
- Mounting of extra block storage

```
. elasticcluster/bin/activate
eval `ssh-agent -s`
ssh-add .ssh/cloud.key
source keystone_rc.sh
elasticcluster -v start slurm -n gridcluster
```

\*Now create and attach the volumes for the sessiondir and cache on the frontend, and ~~var~~ on the compute nodes. See below.\*  
Working on putting this into playbook, however, having problems installing dependencies for the ansible module that does this.

```
elasticcluster -v setup gridcluster -- grid-uh-cloud/ansible/grid_cluster_setup/frontend-local.yml --tags "install-localarc"
elasticcluster -v setup gridcluster -- grid-uh-cloud/ansible/grid_cluster_setup/frontend-local.yml --tags "gridmap,certif,runtime,volumes"
elasticcluster -v setup gridcluster -- grid-uh-cloud/ansible/grid_cluster_setup/computenodes.yml --tags "cvmfs,volumes"
elasticcluster -v setup gridcluster -- grid-uh-cloud/ansible/grid_cluster_setup/common.yml --tags "users,disable_selinux"
```

# Elasticcluster before and after playbooks

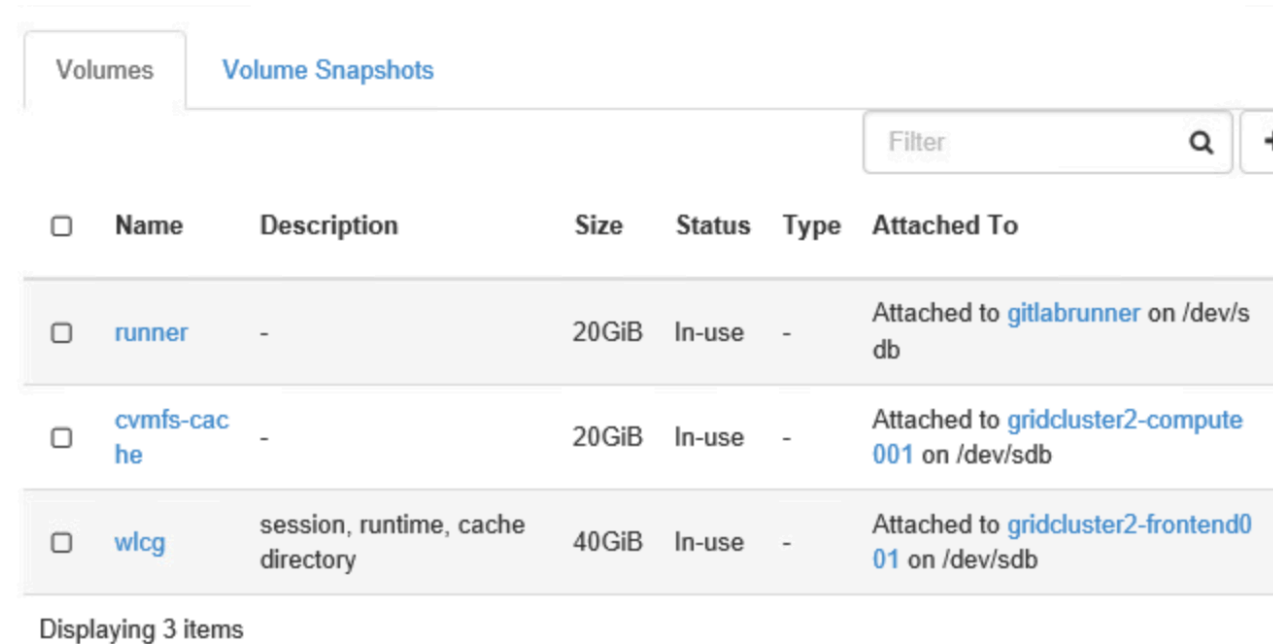
- Used to customize your cluster
- I use the after playbook to
  - Include sessiondir, cache and runtime dirs in NFS
  - Could also use it to create custom slurm user (however do this in my own ansible scripts right now)
- One manual intervention needed: worker nodes and frontend need extra storage volume



# A bit of a hassle: attaching extra volume storage to the instances

The storage space on each cloud machine is too small

- First create volumes in UH IaaS web interface
  - Might be possible to do this using CLI and thus through ansible, but have not prioritized investigating further
- Then attach to the frontend and compute nodes in UH IaaS web interface
  - Could be done in ansible, however had problems getting right python version with this functionality
- Set up filesystem and mountpoint on the machines
  - Done with ansible



The screenshot shows the 'Volumes' tab in the OpenStack dashboard. It features a search filter and a table with columns for Name, Description, Size, Status, Type, and Attached To. Three volumes are listed: 'runner' (20GiB, In-use, attached to gitlabrunner), 'cvmfs-cache' (20GiB, In-use, attached to gridcluster2-compute001), and 'wlcg' (40GiB, In-use, attached to gridcluster2-frontend001). The interface also shows 'Displaying 3 items' at the bottom.

<input type="checkbox"/>	Name	Description	Size	Status	Type	Attached To
<input type="checkbox"/>	runner	-	20GiB	In-use	-	Attached to <a href="#">gitlabrunner</a> on /dev/sdb
<input type="checkbox"/>	cvmfs-cache	-	20GiB	In-use	-	Attached to <a href="#">gridcluster2-compute001</a> on /dev/sdb
<input type="checkbox"/>	wlcg	session, runtime, cache directory	40GiB	In-use	-	Attached to <a href="#">gridcluster2-frontend001</a> on /dev/sdb

Displaying 3 items

# LOCAL submission plugin

# Original plan

- Do not mix ARC client and server classes
- Look at the gridftp jobplugin class and implement something similar for LOCAL submission plugin
  - Jobplugin creates all necessary for job to be created or destroyed
    - Generates jobid
    - Chooses controldir and sessiondir and creates sessiondir
      - Check uploads inputfiles?
    - Controldir files (description, local, proxyfile, status-file)
  - Once these files are inside the controldir a-rex picks up the jobs and processes them, and updates the status file.
- Proof of concept more or less in place before summer, could submit, cancel, kill and get job via LOCAL submission plugin

Then:

Tromsø meeting: this is not a good idea, is copy paste+edit of code.

Don't worry much about client and server methods being mixed.

Use ArexJob class.

# Actual implementation

- Rewritten LOCAL submission plugin to use ArexJob class.
- Simplified things substantially, was more or less rewritten within a few days

Arexjob class basically does what the (gridftp) jobplugin does.

- In LOCALClient: create instance of the ARexJob object, and use the methods directly
  - ARexJob takes care of everything related to creation of job
    - Jobid
    - Generates files for controldir (job.<jobid>.description, job.<jobid>.local, job.<jobid>.proxy)
    - creates, resumes, cancels, kills a job
    - ...

```

bool LOCALClient::submit(const std::list<Arc::JobDescription>& jobdescs, std::list<LOCALJob>& localjobs, const std::string delegation_id)
//called by SubmitterPluginLOCAL ac->submit(..)
logger.msg(Arc::VERBOSE, "Submitting job ");

bool noFailures = true;
int limit = 1000000; // 1 M - Safety
std::list<Arc::JobDescription>::const_iterator itSubmit = jobdescs.begin(), itLastProcessedEnd = jobdescs.begin();
while (itSubmit != jobdescs.end() && limit > 0) {
    for (int i = 0; itSubmit != jobdescs.end() && i < limit; ++itSubmit, ++i) {

        LOCALJob localjob;
        //set some additional parameters
        if(config->DefaultQueue().empty() && (config->Queues().size() == 1)) {
            config->SetDefaultQueue(*(config->Queues().begin()));
        }

        ARex::JobDescriptionHandler job_desc_handler(*config);
        ARex::JobLocalDescription job_desc;
        std::string jobdesc_str;
        Arc::JobDescriptionResult ures = (*itSubmit).UnParse(jobdesc_str, "emies:adl");
        Arc::XMLNode jsdl(jobdesc_str);
        ARex::JobIDGeneratorLOCAL idgenerator(endpoint);
        const std::string dummy = "";

        ARex::ARexJob arexjob(jsdl, *arexconfig, delegation_id, dummy, logger, idgenerator);
        std::cout<<"Maiken-p localclient delegation_id: " << delegation_id<<std::endl;

        if(!arexjob){
            //SET some meaningful error message
            return false;
        }
        else{
            //make localjob for internal handling
            LOCALJob localjob(arexjob, *config, delegation_id);
            /*Set the state of the local job */
            localjob.state = (std::string)arexjob.State();
            localjobs.push_back(localjob);
        }
    }
    itLastProcessedEnd = itSubmit;
}
return noFailures;
}

```

# Configuration of ARC

```
# This file is managed by ansible

# UiO UH-sky arc.conf

[common]
lockfile="/home/centos/lock/arched-arex.lock"
hostname="localhost"
lrms="SLURM"
slurm_wakeupperiod="60"
globus_tcp_port_range="9000,10000"
globus_udp_port_range="9000,10000"
gridmap="/etc/grid-security/grid-mapfile"
shared_filesystem="yes"
tmpdir='${LOCALTMP}'

[vo]
id="local"
vo="local"
file="/etc/grid-security/grid-mapfile"
source="file:///etc/grid-security/grid-mapfile"
mapped_unixid="griduser"

[group/users]
name="users"
vo="TESTVO"
```

```
[grid-manager]
debug="5"

controldir="/home/centos/control"
sessiondir="/wlcg/session"
runtimedir="/wlcg/runtime"
shared_scratch="/wlcg"

lockfile="/home/centos/grid-manager.lock"
pidfile="/home/centos/grid-manager.pid"

logfile="/home/centos/grid-manager.log"
wslogfile="/home/centos/ws-interface.log"
joblog="/home/centos/gm-jobs.log"
logsize="1000000 52"

cachedir="/wlcg/cache"
cachesize="80 70"
cachelifetime="60d"

shared_filesystem="yes"

maxjobs="10000 10000 80"
wakeupperiod="30"
enable_arc_interface="no"
enable_emies_interface="no"
maxtransfertries="10"
```

```
[infosys]
lrms="SLURM"
port="2135"
timelimit="1800"
registrationlog="/home/centos/inforegistration.log"
providerlog="/home/centos/infoprovider.log"
provider_loglevel="2"
```

```
[cluster]
hostname="localhost"
cluster_alias="UIO_CLOUD"
cluster_location="NO-0316"
cluster_owner="University of Oslo (USIT)"
nodeaccess="outbound"
architecture="x86_64"
opsys="CentOS"
```

```
[queue/main]
queue_name="main"
comment="Main queue"
scheduling_policy="FIFO"
comment="UH-cloud LOCAL queue for grid jobs"
architecture="x86_64"
cachetime="90"
timelimit="60"
sizelimit="5000"
totalcpus="1500"
```

# Services needed on ARC-CE for LOCAL submission only

To start ARC-CE with LOCAL submission only

```
service a-rex start
```

Installation performed as local user.

ARC run as local user.

No host certificate required.

# LOCAL plugin in use

LOCAL job submission plugin loaded if

-S org.nordugrid.local submission interface chosen

```
arcsub -d 5 --direct -c 158.39.75.112 -S org.nordugrid.local hello.xrls
```

Or if hostname is set to localhost in arc.conf:

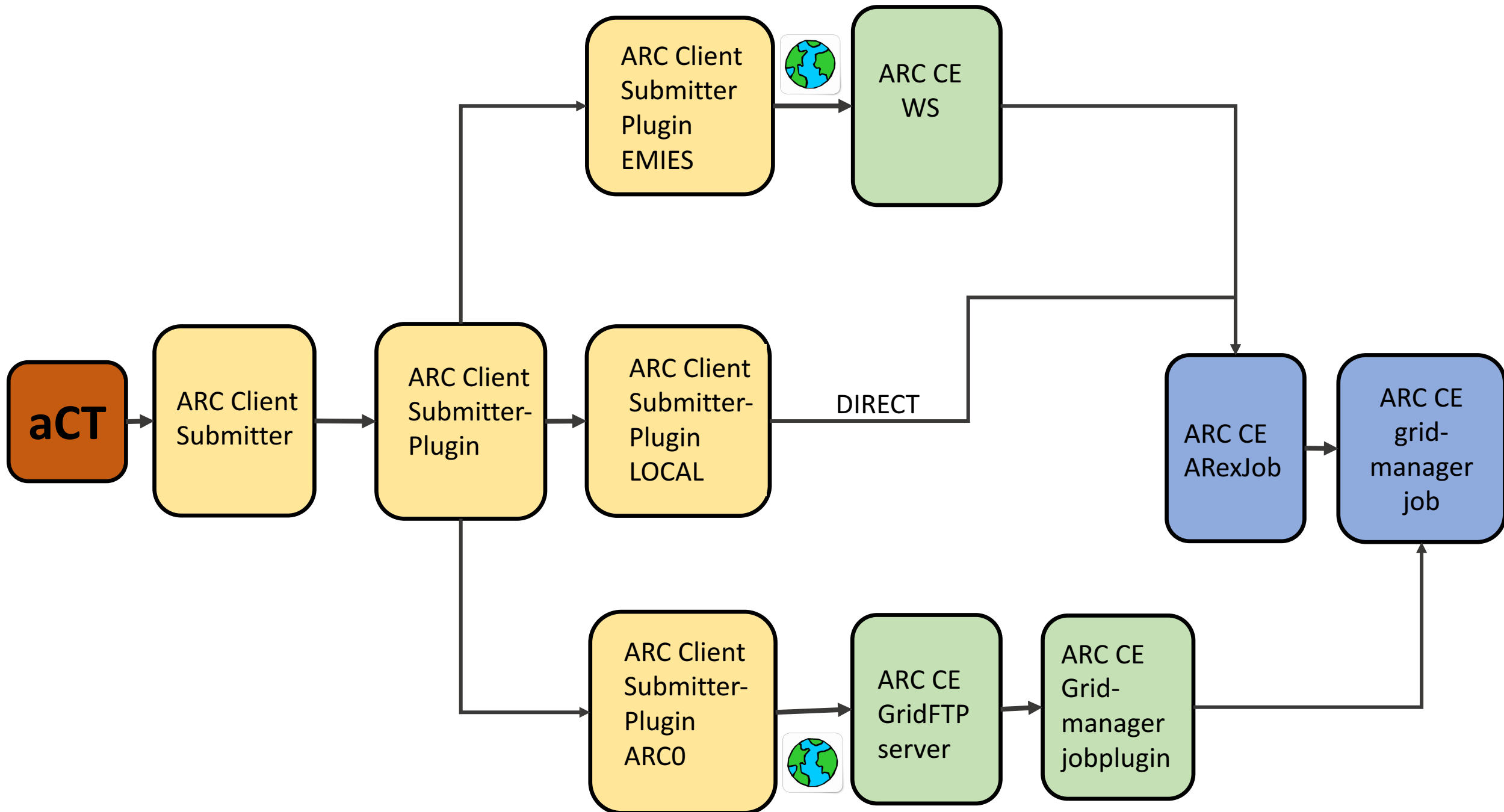
```
arcsub -d 5 --direct -c localhost -S org.nordugrid.local hello.xrls
```

```
[[centos@frontend001 arctestng]$ arcstat -c localhost --long
```

```
Job: local://localhost/q5ZNDmJ4CdrnzfEJwm4kCpGoABFKDmABFKDm6SIKDmABFKDmmXOthn
Name: hello_LOCAL-CLOUD-ARC
State: Finishing
Specific state: FINISHING
ID on service: q5ZNDmJ4CdrnzfEJwm4kCpGoABFKDmABFKDm6SIKDmABFKDmmXOthn
Service information URL: local://localhost (org.nordugrid.local)
Job status URL: local://localhost (org.nordugrid.local)
Job management URL: local://localhost (org.nordugrid.local)
```

Status of 1 jobs was queried, 1 jobs returned information

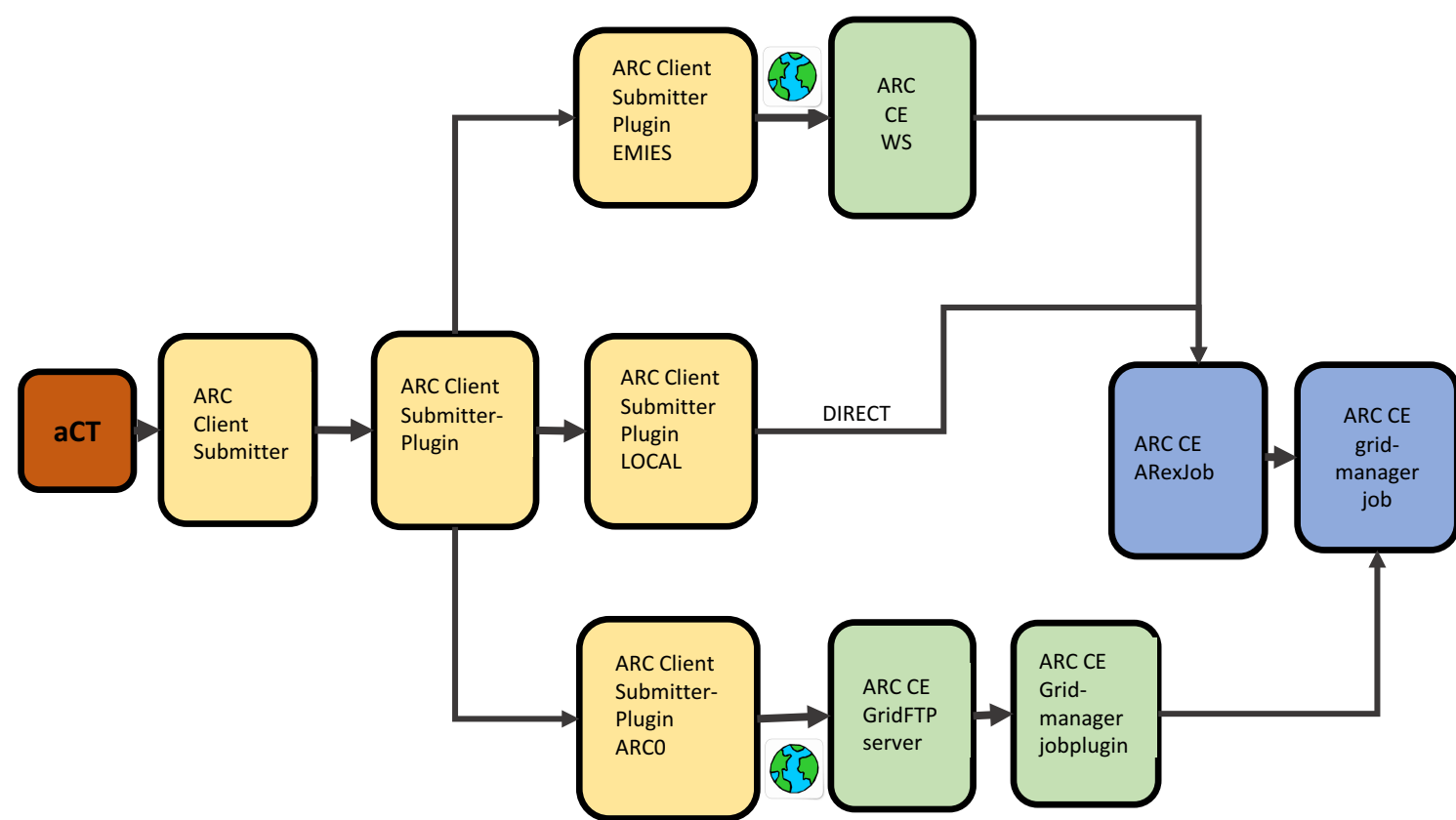




# A LOCAL job

All communication is now directly via file system on the CE, internal memory of the Arex and direct access to the Arex classes and methods

- No layer inbetween like web-service or gridftp server
- Client connects to the chosen JobSubmission plugin and hands over the job description
- SubmissionPluginLOCAL prepares the jobdescription
  - Delegations is sorted out and added to job description if needed for file transfer
- SubmissionPluginLOCAL calls submit method in the LOCALClient which in turn creates an instance of an ArexJob
- In LOCALClient a localjob is created for internal handling (in the same way as an emi-es job is created in the EMI-ES submission plugin)
  - Used for the LOCAL submission plugins internal bookkeeping
- A-Rex picks up job and makes sure it gets processed, and status is updated
- LOCAL submission plugin uses the status file to extract actual state of job and list of jobs in system



# Implementation details

- Source placed under `src/services/a-rex/localjobplugin`
  - Aleksandr helped with proper inclusions and layout for building
- Using grid-manager job states
  - No need for the fine-grained jobstates provided for EMI-ES
- `job.<jobid>.xml` not needed although the infoprovider has been extended to deliver this for the LOCAL jobs
- `Info.xml` does not need information about the LOCAL submission service, since this is not visible from outside.
  - However, the infoprovider has been extended to provide this if needed/we decide to publish some information

# Changes in inforprovider

- Florido added the local submission endpoint
- Jobstate mapping for LOCAL plugin done
- Info.xml and job.jobid.xml contains information about the local submission endpoint and jobs submitted via local submission interface
- Healthstate not dependent on host certificate being in place
  - LOCAL submission plugin does not require host certificate

```
sub local_state {
  ## Maps the gm_state added with detailed info of the lrms_state to local-state
  ## Added the local state terminal to gm_state KILLED - this will allow the job to be cleaned when calling arcclean
  #require Data::Dumper; import Data::Dumper qw(Dumper);

  # TODO: probably add $failure_state taken from somewhere
  my ($gm_state, $lrms_state, $failure_state) = @_;
  my $loc_state = {
    'state' => ''
  };
};
```

```
## NorduGrid local submission
my $getNorduGridLocalSubmissionEndpoint = sub {
  # don't publish if no endpoint URL
  #return undef unless $config->(enable_emies_interface);
  # To-decide: should really the local submission plugin be present in info.xml? It is not useful for the outside world.
  my $cep = {};

  $cep->(CreationTime) = $creation_time;
  $cep->(Validity) = $validity_ttl;

  $cep->(ID) = "$NGLScepIDP";

  $cep->(Name) = "ARC CE Local Submission";

  # OBS: ideally HED should be asked for the URL
  $cep->(URL) = $config->(endpoint);
  # TODO: define a strategy to add data capabilities
  # $cep->(Capability) = $epscapabilities->{'org.ogf.glue.emies.activitycreation'};
  $cep->(Technology) = 'direct';
  $cep->(InterfaceName) = 'org.nordugrid.local';
  $cep->(InterfaceVersion) = [ '1.0' ];
  $cep->(Capability) = [ @$epscapabilities->{'org.nordugrid.local'}, @$epscapabilities->{'common'} ];
  $cep->(Implementor) = "NorduGrid";
  $cep->(ImplementationName) = "nordugrid-arc";
  $cep->(ImplementationVersion) = $config->(altversion);

  $cep->(QualityLevel) = "testing";

  my %healthissues;
  # Host certificate not required for LOCAL submission interface.
  if ( $host_info->(gm_alive) ne 'all' ) {
    if ( $host_info->(gm_alive) eq 'some' ) {
      push @{$healthissues{warning}}, 'One or more grid managers are down';
    } else {
      push @{$healthissues{critical}},
        $config->(remotegmdirs) ? 'All grid managers are down'
          : 'Grid manager is down';
    }
  }

  if (%healthissues) {
    my @infos;
    for my $level (qw(critical warning other)) {
      next unless $healthissues{$level};
      $cep->(HealthState) ||= $level;
      push @infos, @{$healthissues{$level}};
    }
    $cep->(HealthStateInfo) = join "; ", @infos;
  } else {
    $cep->(HealthState) = 'ok';
  }
};
```

```
$loc_state->(State) = [ "finished","failure" ];
return $loc_state;
} elsif ($failure_state eq "DELETED") {
  $loc_state->(State) = [ "deleted","failure" ];
  return $loc_state;
} elsif ($failure_state eq "CANCELING") {
  $loc_state->(State) = [ "canceling","failure" ];
  return $loc_state;
} else {
  return $loc_state;
}
} elsif ($gm_state eq "FINISHED") {
  $loc_state->(State) = [ "finished" ];
};
```

```
/// \mapattr ACCEPTED -> ACCEPTED
/// \mapattr ACCEPTING -> ACCEPTED
if ((state_ == "accepted") ||
    (state_ == "accepting"))
  return JobState::ACCEPTED;
/// \mapattr PREPARING -> PREPARING
/// \mapattr PREPARED -> PREPARING
```

```
paring"} ||
pared")) ||
RING;
UBMITTING
-> SUBMITTING
mit") ||
mitting"))
TING;
QUEUING
ms:q")
NG;
RUNNING
ms:r")
NG;
HOLD
ms:h")

HOLD
ms:s")

FINISHING
ms:e")
HING;
HOLD
ms:o")

QUEUING
,6) == "inlrms")
NG; // expect worst?
> FINISHING
FINISHING
> FINISHING
FINISHING
ishing"} ||
ling"} ||
celing"} ||
cuted"))
HING;
FINISHED
shed")
HED;
ILLED
ed")
D;
ILLED
ed")
D;
DELETED
ted")
```

```
return JobState::DELETED;
/// \mapattr "" -> UNDEFINED
else if (state_ == "")
  return JobState::UNDEFINED;
/// \mapattr Any other state -> OTHER
else
  return JobState::OTHER;
};
```

aCT

```

<config>
<db>
  <type>mysql</type>
  <name>act</name>
  <user>centos</user>
  <password>xxxxxxx</password>
  <host>localhost</host>
  <port>3306</port>
</db>
<loop>
  <periodicrestart>
    <actsubmitter>120</actsubmitter>
    <actstatus>600</actstatus>
    <actfetcher>600</actfetcher>
    <actcleaner>600</actcleaner>
  </periodicrestart>
</loop>
<tmp>
  <dir>/home/centos/software/aCT/act-test/tmp</dir>
</tmp>
<actlocation>
  <dir>/home/centos/software/aCT/src</dir>
  <pidfile>/home/centos/software/aCT/act-test/act.pid</pidfile>
</actlocation>
<logger>
  <level>debug</level>
  <arcllevel>debug</arcllevel>
  <logdir>/home/centos/software/aCT/act-test/log</logdir>
  <rotate>25</rotate>
</logger>
<atlasgiis>
  <timeout>20</timeout>
</atlasgiis>

```

```

<voms>
  <vo>atlas</vo>
  <roles>
    <item>production</item>
    <item>pilot</item>
  </roles>
  <bindir>/home/centos/software/bin</bindir>
  <proxylifetime>345600</proxylifetime>
  <minlifetime>259200</minlifetime>
  <proxypath>/home/centos/software/aCT/proxies/atlact1.rfc.long.proxy</proxypath>
  <cacertdir>/etc/grid-security/certificates</cacertdir>
  <proxystoredir>/home/centos/software/aCT/proxies</proxystoredir>
</voms>

```

# Configuration of aCT

```
<config>
<executable>
  <wrapperurl>http://www-f9.ijs.si;cache=check/grid/ARCPilot-test</wrapperurl>
  <ptarurl>http://pandaserver.cern.ch:25080;cache=check/cache/pilot/pilotcode.tar.gz</ptarurl>
  <ptarurlrc>"http://project-atlas-gmsb.web.cern.ch;cache=no/project-atlas-gmsb/pilotcode-rc.tar.gz"</ptarurlrc>
</executable>

<joblog>
  <urlprefix>http://158.39.75.112/act/jobs</urlprefix>
  <dir>/var/www/html/act/jobs</dir>
</joblog>

<agis>
  <server>http://atlas-agis-api.cern.ch/request/pandaqueue/query/list/&#63;json&#38;preset&#61;schedconf.all</server>
  <objectstores>http://atlas-agis-api.cern.ch/request/ddmendpoint/query/list/&#63;json&#38;type&#91;&#93;&#61;OS_LOGS&#38;type&#91;&#93;&#61;OS_ES</objectstores>
  <jsonfilename>/home/centos/software/aCT/tmp/agis.json</jsonfilename>
  <osfilename>/home/centos/software/aCT/tmp/oses.json</osfilename>
  <pilotmanager>aCT</pilotmanager>
  <maxjobs>0</maxjobs>
</agis>

<panda>
  <server>https://pandaserver.cern.ch:25443/server/panda/</server>
  <heartbeattime>1800</heartbeattime>
  <threads>1</threads>
  <getjobs>1</getjobs>
  <schedulerid>aCT-uo-cloud</schedulerid>
  <timeout>60</timeout>
  <minjobs>10</minjobs>
</panda>
</config>
```

```
<panda>
  <server>https://pandaserver.cern.ch:25443/server/panda/</server>
  <heartbeattime>1800</heartbeattime>
  <threads>1</threads>
  <getjobs>1</getjobs>
  <schedulerid>aCT-uo-cloud</schedulerid>
  <timeout>60</timeout>
  <minjobs>10</minjobs>

  <sites>
    <site>
      <name>UIO_CLOUD</name>
      <endpoints>
        <item>local://158.39.75.112:/main</item>
      </endpoints>
      <maxjobs>2</maxjobs>
      <corecount>1</corecount>
    </site>
  </sites>
</panda>
</config>
```

# Status:

- Can use `addnewjob.py` to insert new job directly into the arc-table for the localjob
- At the moment setting up aCT to receive Hammercloud jobs



# .... what more ...

- At the moment installing from source
- Packaging should be sorted out for LOCAL plugin
  - Not complete overview of whether it is already fine or not (ldap should already have been sorted out, also want to install w/o gridftp)
  - And: how to install rpms as local user
- Will continue to check through code. Some solutions might need to be cleaned up. Some might need to be improved.