GLEXEC, JOBS AND CATALOGUES

Federico Carminati
ALICE
GDB, June 8, 2011

QUESTIONS

- Security (actually rather should be authentication and authorisation):
  - gLExec. can we do this in a much simpler way by trusting the experiment frameworks? What might simpler alternatives be?
  - What are the real needs for file access protection? What is really needed? What is the simplest way to implement what is needed?
  - Input to the Identity Federation Workshop. Is the use of X509 currently an issue for the experiments?
- Job management:
  - Pilot jobs are (almost) ubiquitous now. What is left that still needs a WMS?
  - Can we simplify the needs for sites – I.e. reduce the complexity of a CE? Do we still need to require the CE to pass parameters to the batch system? Don't your pilot frameworks do all this anyway?
  - What is the intent with pilot factories? Will they be deployed at sites? If so, surely that replaces the CE completely (apart from a trivial mechanism to launch the factory at a site).

GLEXEC@ALICE

- The AliEn JobAgent was enabled to use gLExec
  - Tests were successful (with manual user proxy delivery)
- Open Issue
  - How to get user proxy from client to WN? (glEXec itself does not provide any solution for this)
- We see three original motivations to use gLExec:
  - Mutually isolate or jail the actual jobs on WN
  - Hence protect pilot (+ pilot proxy!) from actual job
  - Allow for on-site accounting, directly and without relying on the VO
- First two points are solved by gLExec, but ...

GLEXEC OPEN ISSUES

- If the VO (Central Services) stores and handles user proxies, it is liable in case they are mixed up or stolen
  - Therefore no benefit in Accountability+Trust with respect to the currently deployed scenario
  - Even worse, if an attacker gets hold of a user proxy from CS, this would be the ideal identity theft
- Using a key (stored and handled in the CS) and putting the user proxies on the MyProxy Server is neither a solution. Who has the key, gets the proxy...
- Finally, how should a user proxy proof that a certain user submitted the actual job at hand?
  - This would be necessary for a meaningful accounting.

CURRENT WORK

- Let a user sign the JDL upon Job submission using its Grid certificate. Send signature plus user certificate (public part) with the JDL to the WN.
  - gLExec would need to verify the signature, and by that ensure THIS job was submitted by the user
  - Verification is analogue to the one of a user proxy, using the public part of the user's certificate that is enclosed
  - Ensure there is no alteration or mix-up of job vs. user and thereby allow for actual accountability
- Allow to limit potential damage to the minimum, following the security principle of least privilege
  - Proposed and currently discussed with the gLExec developers

FILE PERMISSIONS

- File permissions are stored in the catalogue
  - UNIX-style permissions
  - ACLs are supported but not used in practice
- Storage has no knowledge of the users and/or permissions
  - Each storage interaction requires a ticket signed by the central services containing
    - Operation (read, write, delete)
    - LFN, PFN, unique identifier, SE name
    - Size, MD5 checksum

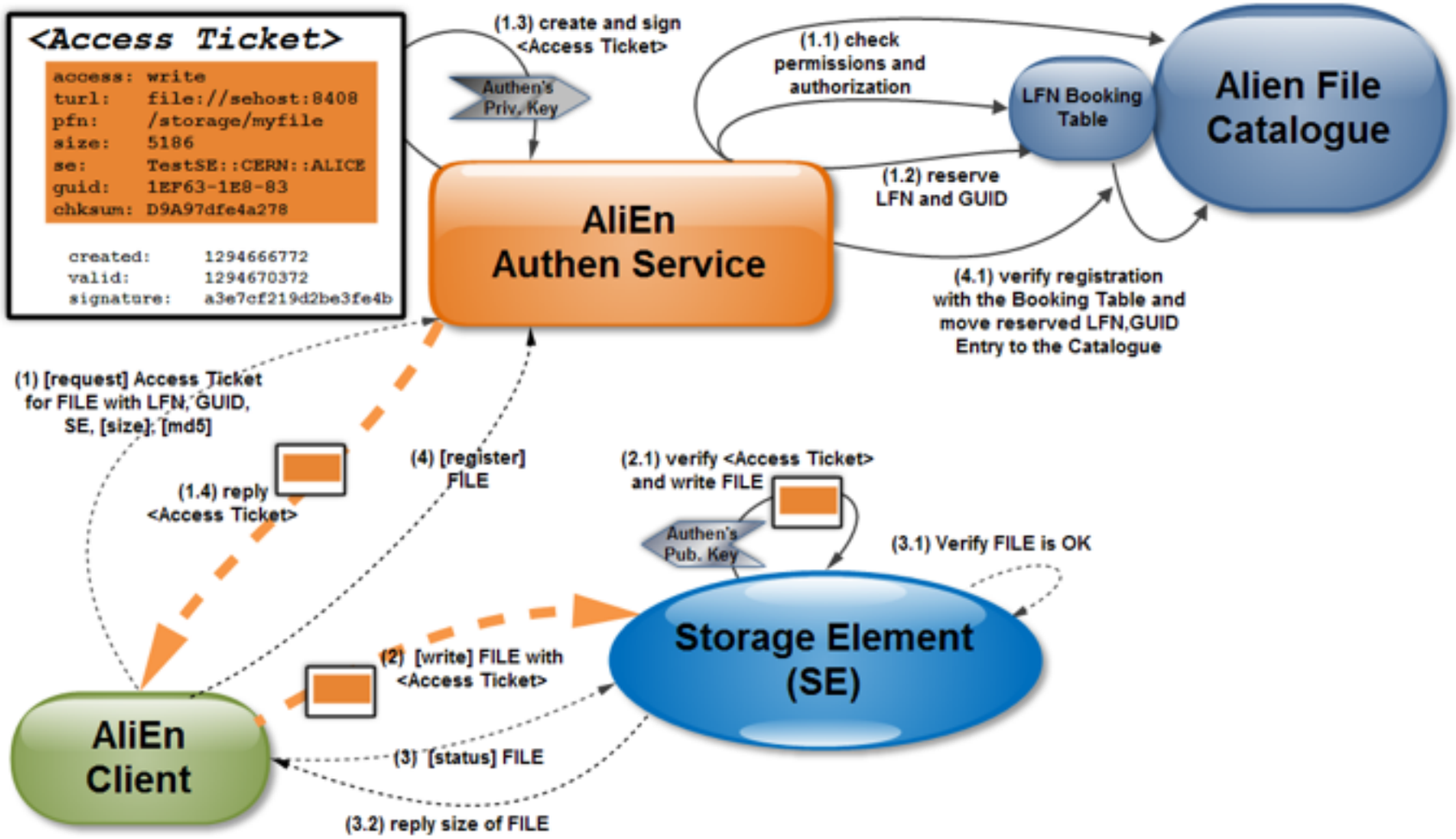
NEW "AUTHENTIC" WRITE OPERATIONS

- Now in regression testing
- First step as before
  - Asking for a write ticket and executing the operation
- To update the catalogue the client will have to present a feedback ticket from the storage
  - File details as the server has seen them
    - PFN, size, checksum
  - Central Services verify that
    - Client has previously asked to write that file
    - Booked details match the storage-provided values
  - Only then the file is committed to the catalogue

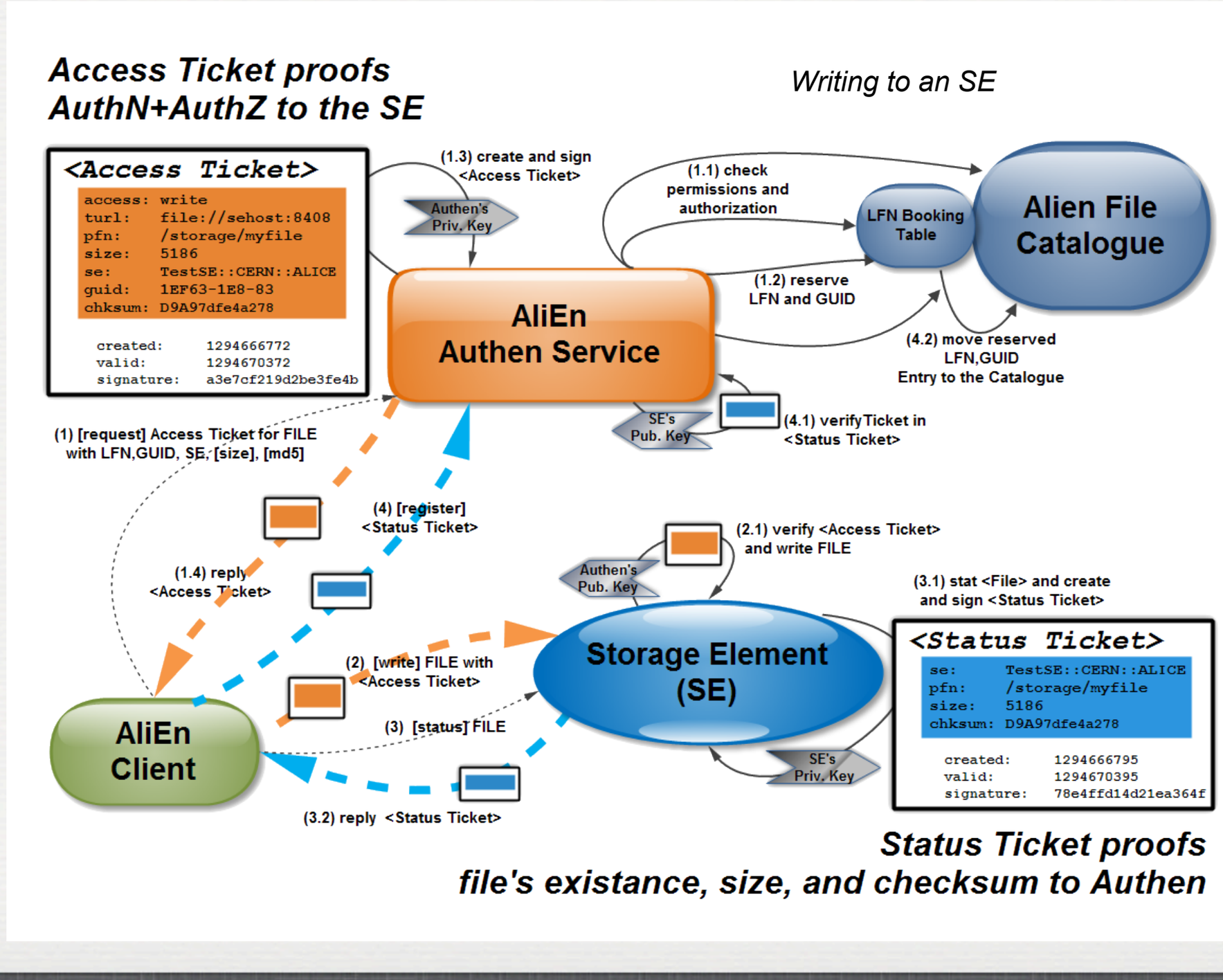
PLANNED IMPLEMENTATION

Access Ticket proofs AuthN+AuthZ to the SE

Writing to an SE

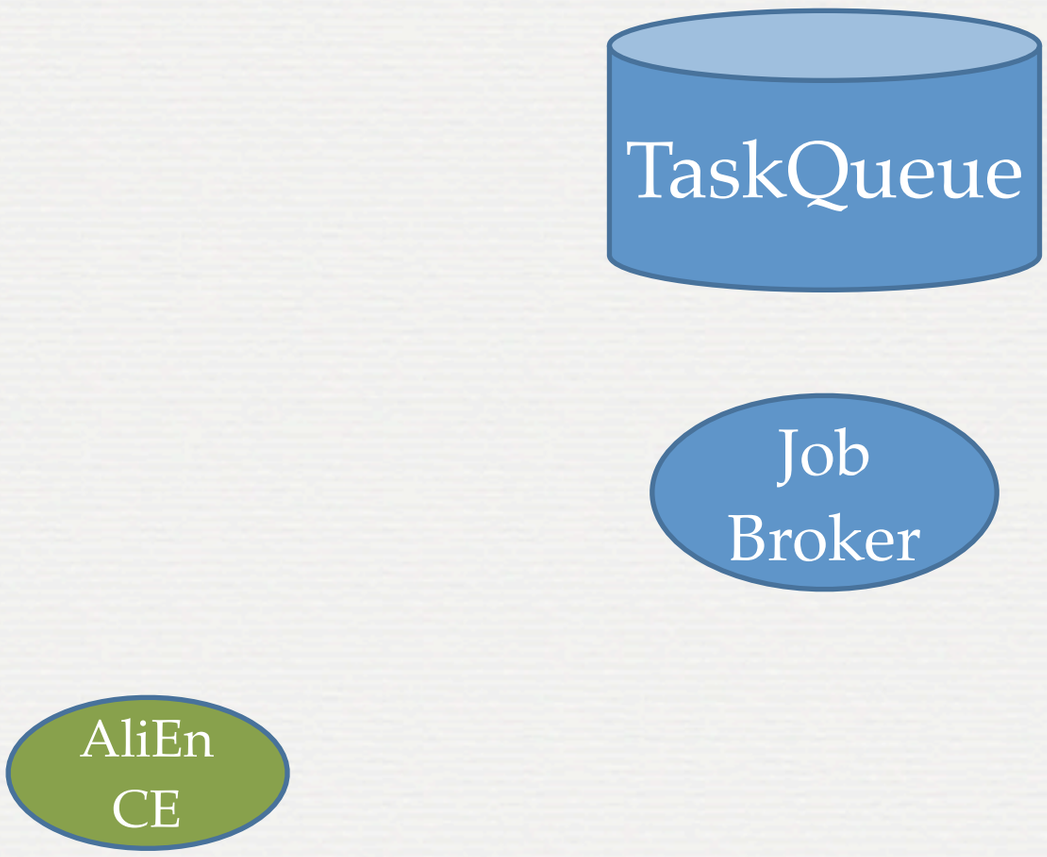


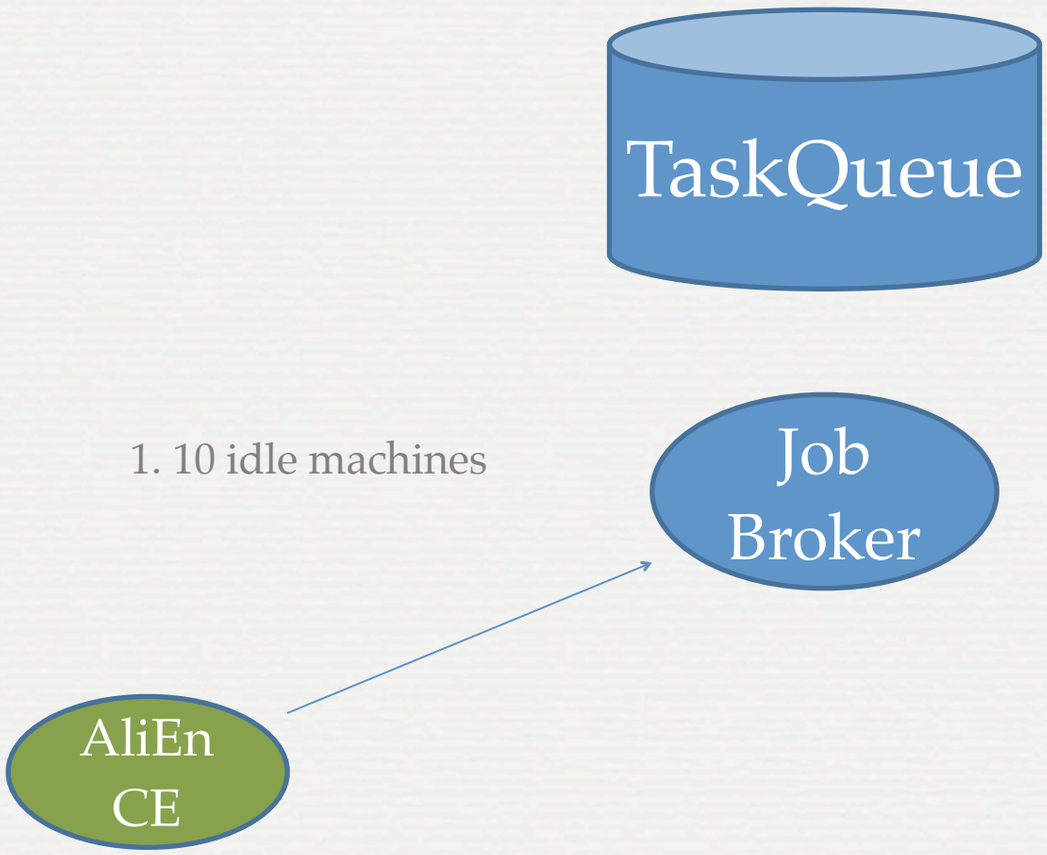
PLANNED IMPLEMENTATION

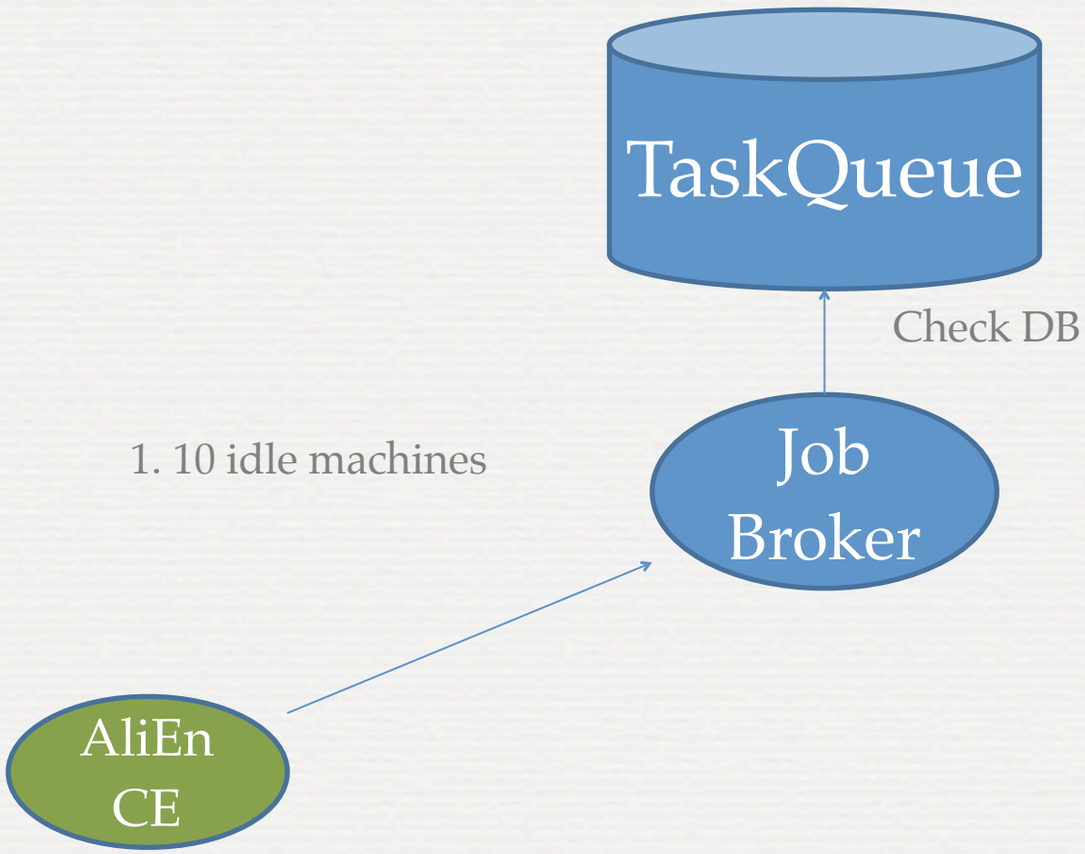


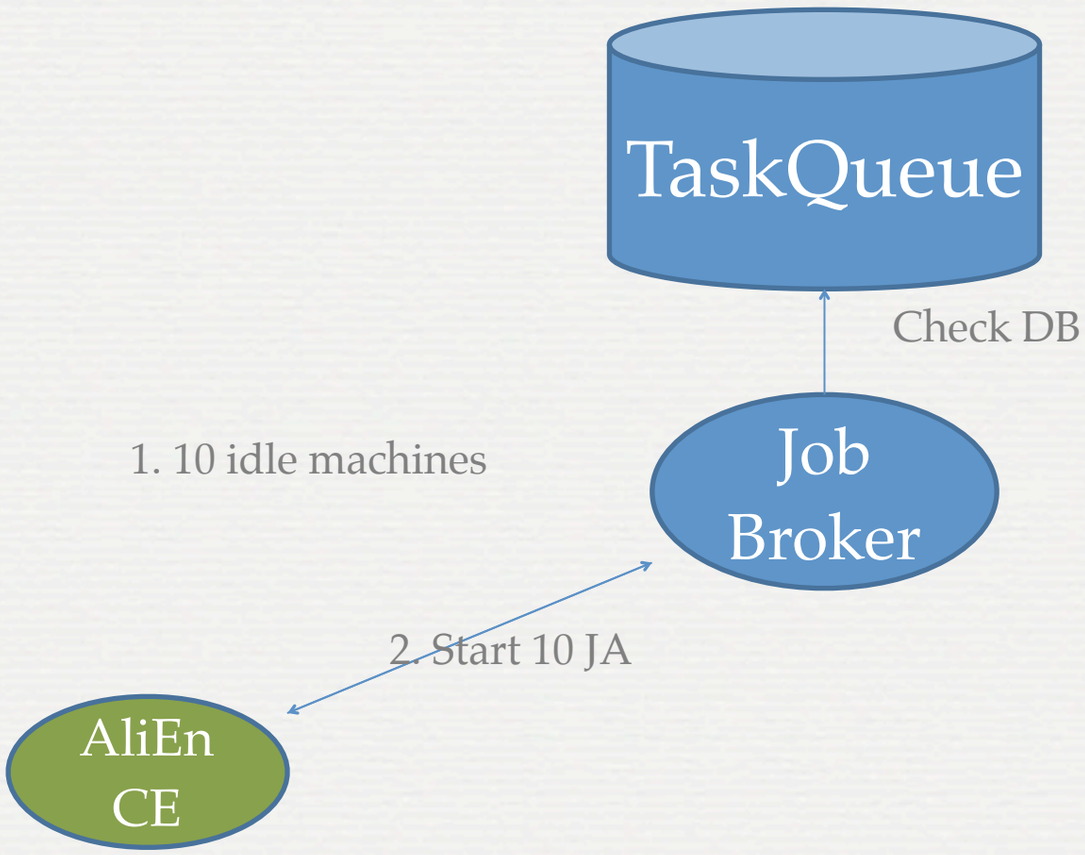
ALIEN JOB BROKERING

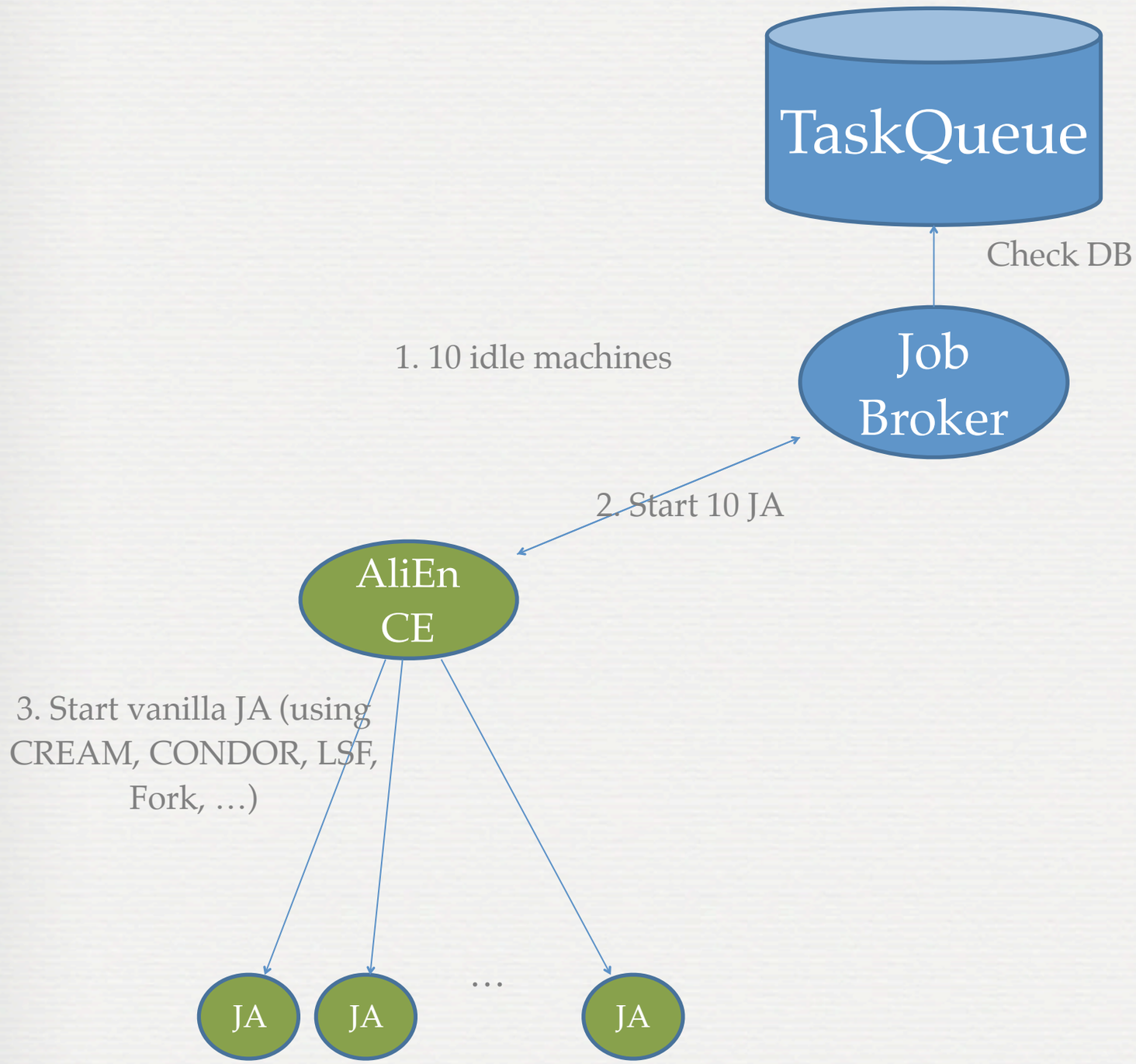
- One single TaskQueue with all the jobs
  - Priorities and quotas per user
- Multiple options for requirements
  - Data, packages, TTL, disk space, grid partitions, user defined…
- Two level brokering
  - CE: advertise available resources and submits vanilla JA
  - JA: Check worker node and gets payload

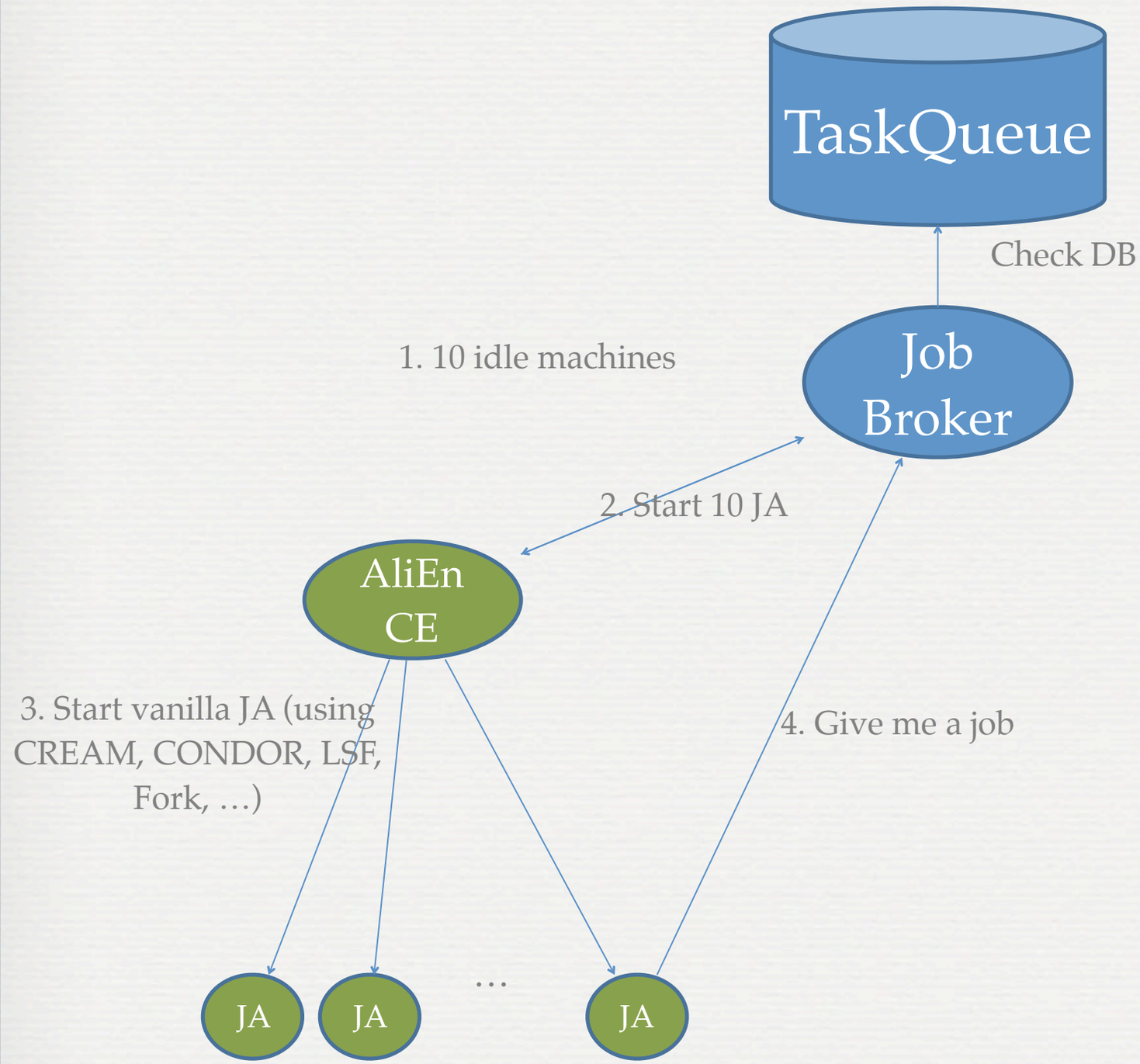

TaskQueue
Job
Broker
AliEn
CE

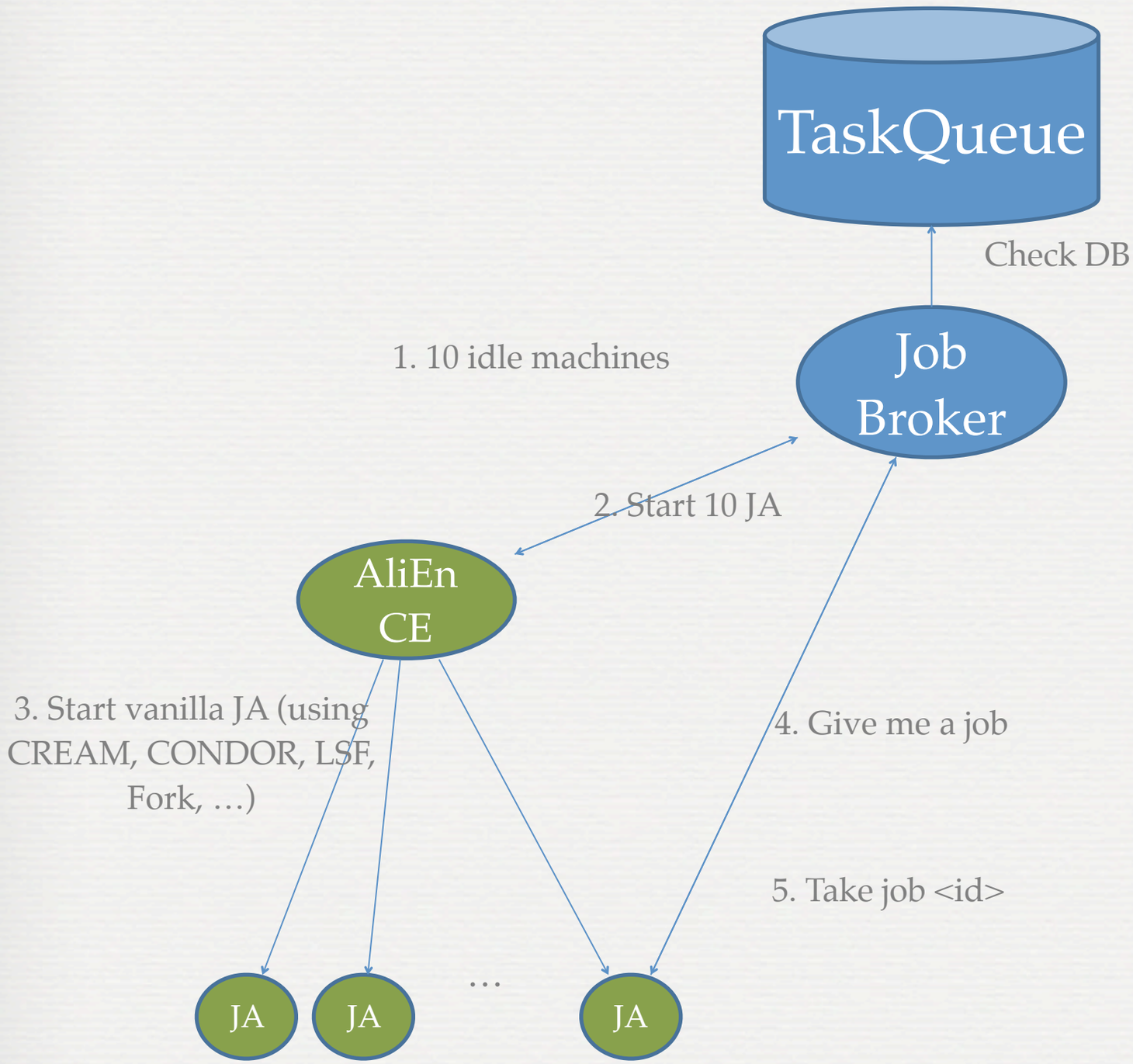

TaskQueue
1. 10 idle machines
Job Broker
AliEn CE

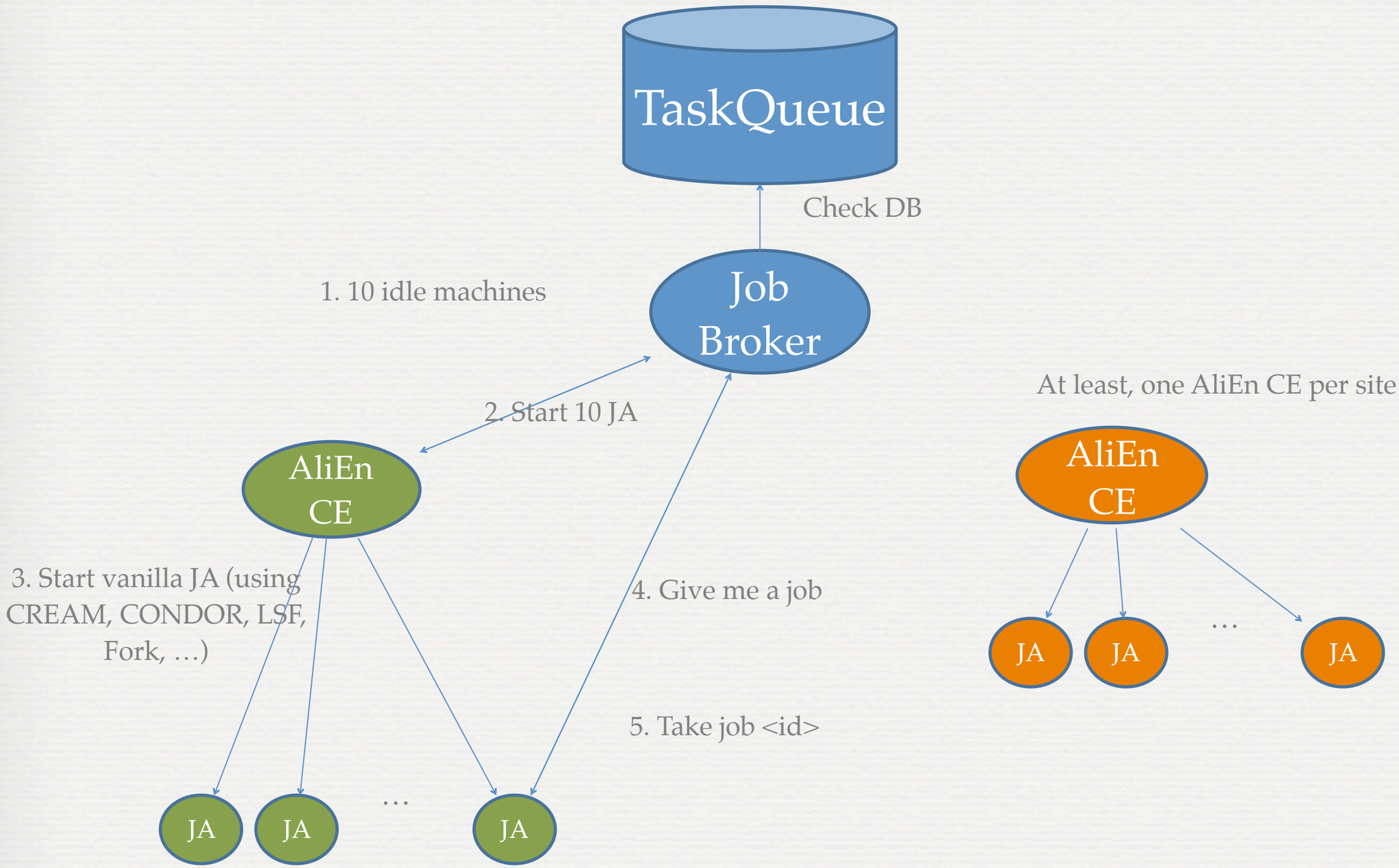

TaskQueue
Check DB
1. 10 idle machines
Job
Broker
AliEn
CE


TaskQueue
Check DB
1. 10 idle machines
Job Broker
2. Start 10 JA
AliEn CE


TaskQueue
Check DB
1. 10 idle machines
Job Broker
2. Start 10 JA
AliEn CE
3. Start vanilla JA (using CREAM, CONDOR, LSF, Fork, …)
JA
JA
…
JA


TaskQueue
Check DB
1. 10 idle machines
Job Broker
2. Start 10 JA
AliEn CE
3. Start vanilla JA (using CREAM, CONDOR, LSF, Fork, …)
4. Give me a job
JA
JA
…
JA


TaskQueue
Check DB
1. 10 idle machines
Job Broker
2. Start 10 JA
AliEn CE
3. Start vanilla JA (using CREAM, CONDOR, LSF, Fork, …)
4. Give me a job
5. Take job <id>
JA
JA
…
JA


TaskQueue
Check DB
1. 10 idle machines
Job Broker
2. Start 10 JA
AliEn CE
3. Start vanilla JA (using CREAM, CONDOR, LSF, Fork, …)
4. Give me a job
5. Take job <id>
JA
JA
…
JA
At least, one AliEn CE per site
AliEn CE
JA
JA
…
JA

USING MULTICORES

- Under development
- Two approaches:
  - Run one JA per core
    - Easy to implement,
  - One JA per machine
    - Detect # cores, and requests multiple payloads.

QUESTIONS

- Security (actually rather should be authentication and authorisation):
  - gLExec. can we do this in a much simpler way by trusting the experiment frameworks? What might simpler alternatives be?
    - We have currently testing a relatively simple way to use gLExec

QUESTIONS

- Security (actually rather should be authentication and authorisation):
  - gLExec. can we do this in a much simpler way by trusting the experiment frameworks? What might simpler alternatives be?
  - What are the real needs for file access protection? What is really needed? What is the simplest way to implement what is needed?
    - We have all the protections implemented in our catalogue. We have no additional requirements

QUESTIONS

- Security (actually rather should be authentication and authorisation):
  - gLExec. can we do this in a much simpler way by trusting the experiment frameworks? What might simpler alternatives be?
  - What are the real needs for file access protection? What is really needed? What is the simplest way to implement what is needed?
  - Input to the Identity Federation Workshop. Is the use of X509 currently an issue for the experiments?
    - No. We will follow the workshop tomorrow.

QUESTIONS

- Security (actually rather should be authentication and authorisation):
  - gLExec. can we do this in a much simpler way by trusting the experiment frameworks? What might simpler alternatives be?
  - What are the real needs for file access protection? What is really needed? What is the simplest way to implement what is needed?
  - Input to the Identity Federation Workshop. Is the use of X509 currently an issue for the experiments?
- Job management:
  - Pilot jobs are (almost) ubiquitous now. What is left that still needs a WMS?
    - Nothing for ALICE

QUESTIONS

- Security (actually rather should be authentication and authorisation):
  - gLExec. can we do this in a much simpler way by trusting the experiment frameworks? What might simpler alternatives be?
  - What are the real needs for file access protection? What is really needed? What is the simplest way to implement what is needed?
  - Input to the Identity Federation Workshop. Is the use of X509 currently an issue for the experiments?
- Job management:
  - Pilot jobs are (almost) ubiquitous now. What is left that still needs a WMS?
  - Can we simplify the needs for sites – I.e. reduce the complexity of a CE? Do we still need to require the CE to pass parameters to the batch system? Don't your pilot frameworks do all this anyway?
    - Yes, we can reduce CE's complexity and we do not need to pass parameters to the batch system. We could also do without CE completely.

QUESTIONS

- Security (actually rather should be authentication and authorisation):
  - gLExec. can we do this in a much simpler way by trusting the experiment frameworks? What might simpler alternatives be?
  - What are the real needs for file access protection? What is really needed? What is the simplest way to implement what is needed?
  - Input to the Identity Federation Workshop. Is the use of X509 currently an issue for the experiments?
- Job management:
  - Pilot jobs are (almost) ubiquitous now. What is left that still needs a WMS?
  - Can we simplify the needs for sites – I.e. reduce the complexity of a CE? Do we still need to require the CE to pass parameters to the batch system? Don't your pilot frameworks do all this anyway?
  - What is the intent with pilot factories? Will they be deployed at sites? If so, surely that replaces the CE completely (apart from a trivial mechanism to launch the factory at a site).
    - We still see the need for simple CEs, we are not using pilot job factories


"That's
all
folks!"