Hortonworks

Secrets of YARN Application Development

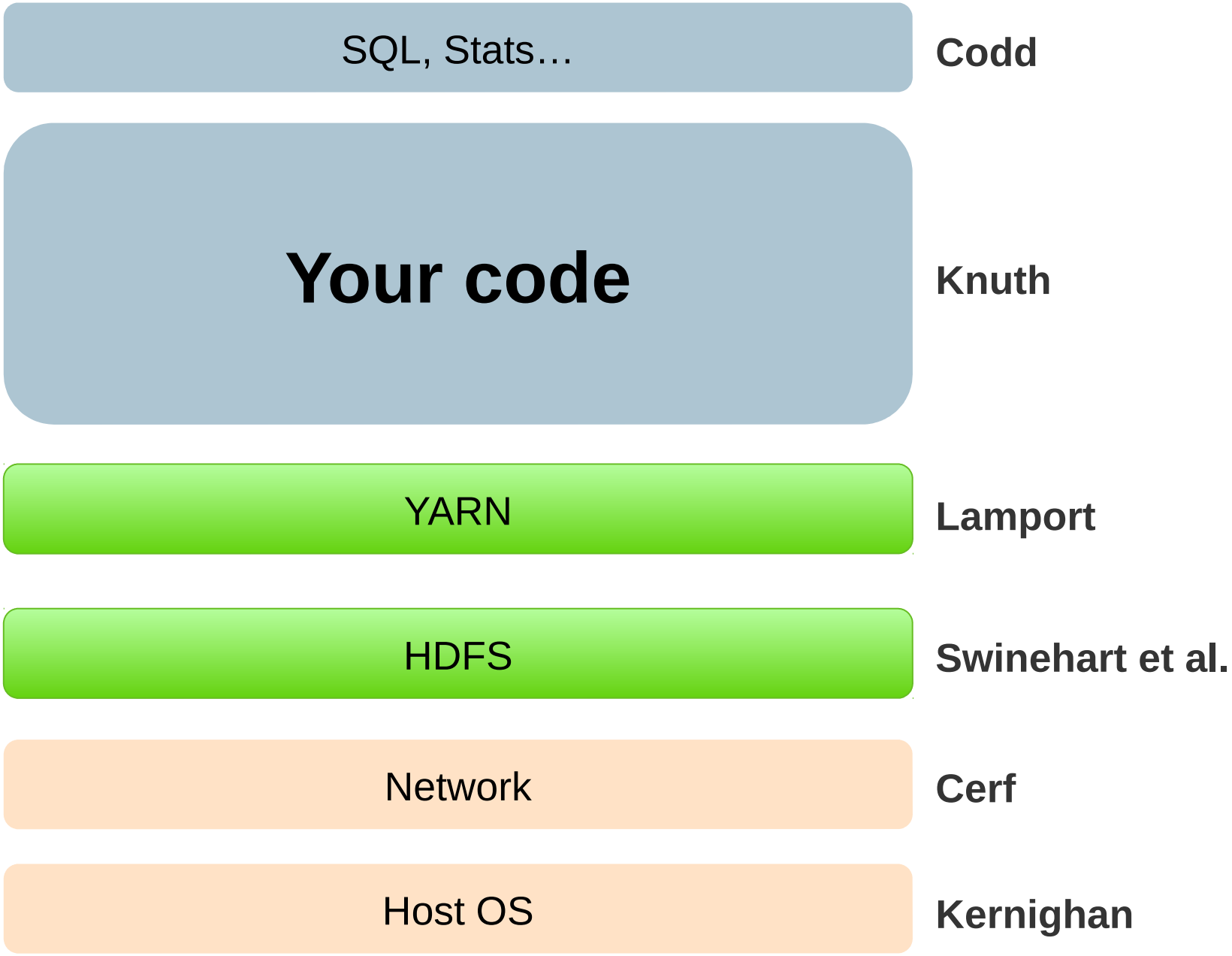
Steve Loughran
stevel at hortonworks.com
@steveloughran

Berlin, May 2014


If all you have is a JobTracker…

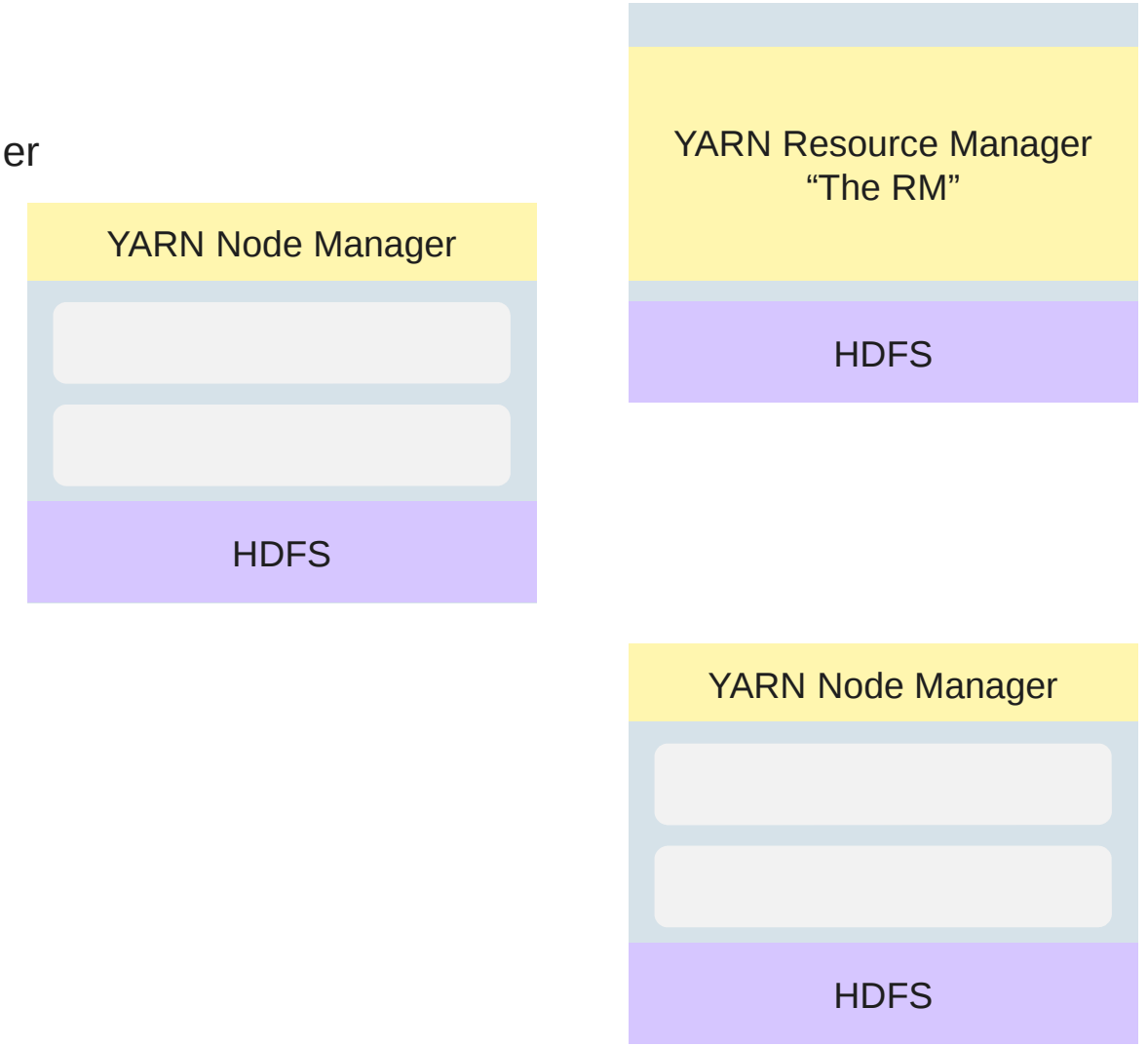
Everything pretends to be an MR Job

YARN:
more tools for the engineers

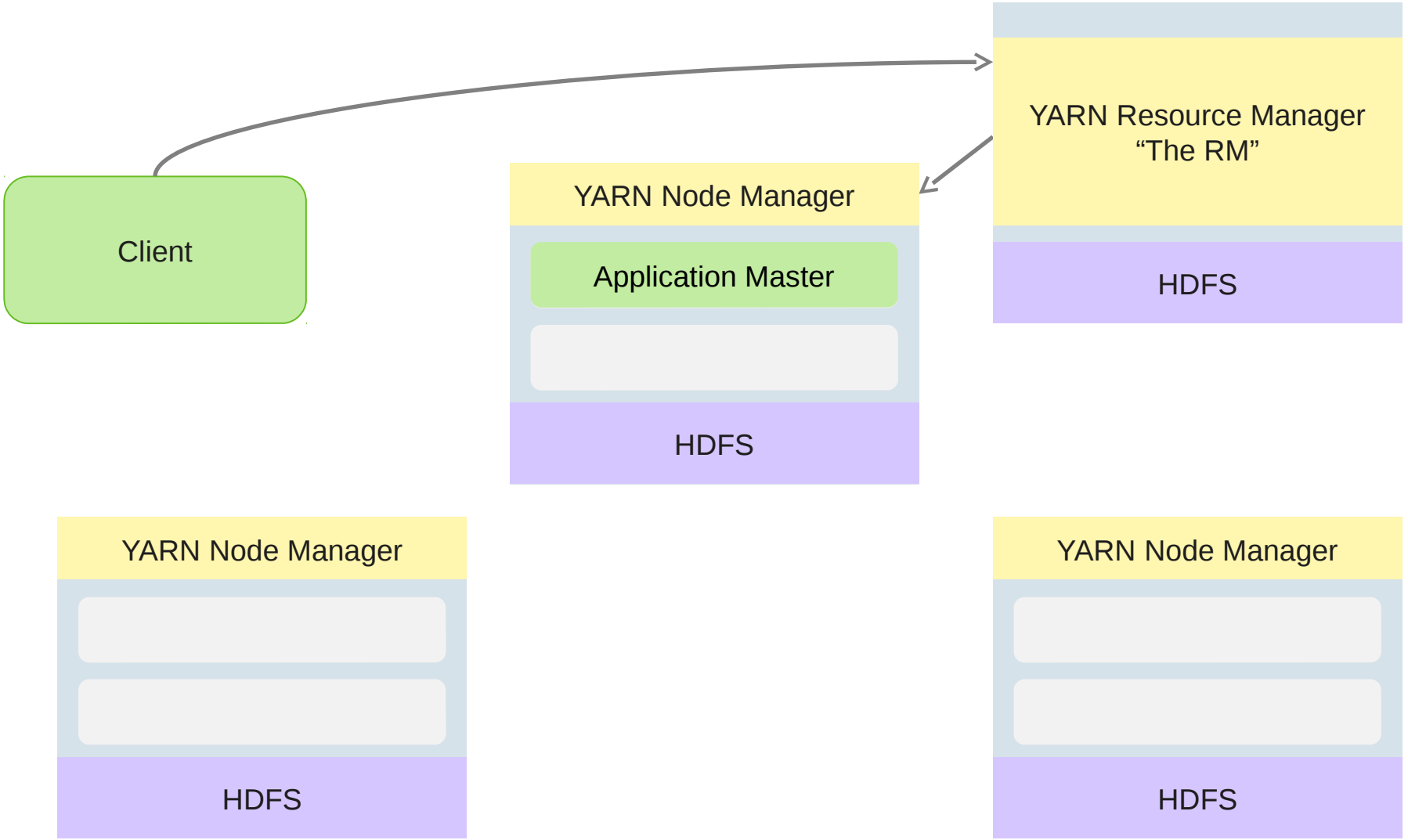


YARN runs code across the cluster

- Servers run YARN Node Managers
- NM's heartbeat to Resource Manager
- RM schedules work over cluster
- RM allocates containers to apps
- NMs start containers
- NMs report container health



Client creates App Master



ContainerLaunchContext

```
// env vars, with env-variables
// ApplicationConstants.Environment.LOG_DIRS.$()
setEnvironment(Map<String,String> env)

// Local resources are files in HDFS, S3 to copy/[+unzip/untar]
setLocalResources(Map<String,LocalResource> localResources)

// bash -c commands

setCommands(List<String> commands);
```

Local Resources pulled from HDFS

```
Map<String,LocalResource> localResources = new HashMap<>();

LocalResource res = Records.newRecord(LocalResource.class);
res.setType(LocalResourceType.ARCHIVE);
res.setVisibility(LocalResourceVisibility.APPLICATION);
URL yamURL
  URL.newInstance("s3n","packages","80","hbase.tar.gz");
res.setResource(yamURL);
resources.put("lib/hbase",res);
ctx.setLocalResources(resources);
```

Classpath setup…

```
public static String buildCP(Configuration conf) {
  StringBuilder cp = new StringBuilder(
     ApplicationConstants.Environment.CLASSPATH.$());

  String[] ycp = conf.getStrings(
     "yarn.application.classpath",
     DEFAULT_YARN_CROSS_PLATFORM_APPLICATION_CLASSPATH);
  List<String> ycpl= Arrays.asList(ycp);
  ycpl.stream().forEach(
     (e) -> cp.append(e).append(':')
  );
  return cp.toString();
 }
```

Classpath Grief

1. Client uploads dependencies to HDFS, adds as resources then onto classpath
2. **"yarn.application.classpath"**
3. **YarnConfiguration. DEFAULT_YARN_CROSS_PLATFORM_ APPLICATION_CLASSPATH**
4. Your code had better use the same JARs as Hadoop
5. HADOOP-9991 "roll up JARs to latest versions"

Better: OSGi, leaner Hadoop client libs

Tasks of Application Master

- Request containers from YARN Resource Manager
- Process allocation responses
- Submit launch requests to run code in containers
- Handle IPC calls from clients and container-side code
- Handle notifications from YARN Resource Manager
- Implement placement policy
- Implement failure handling policy

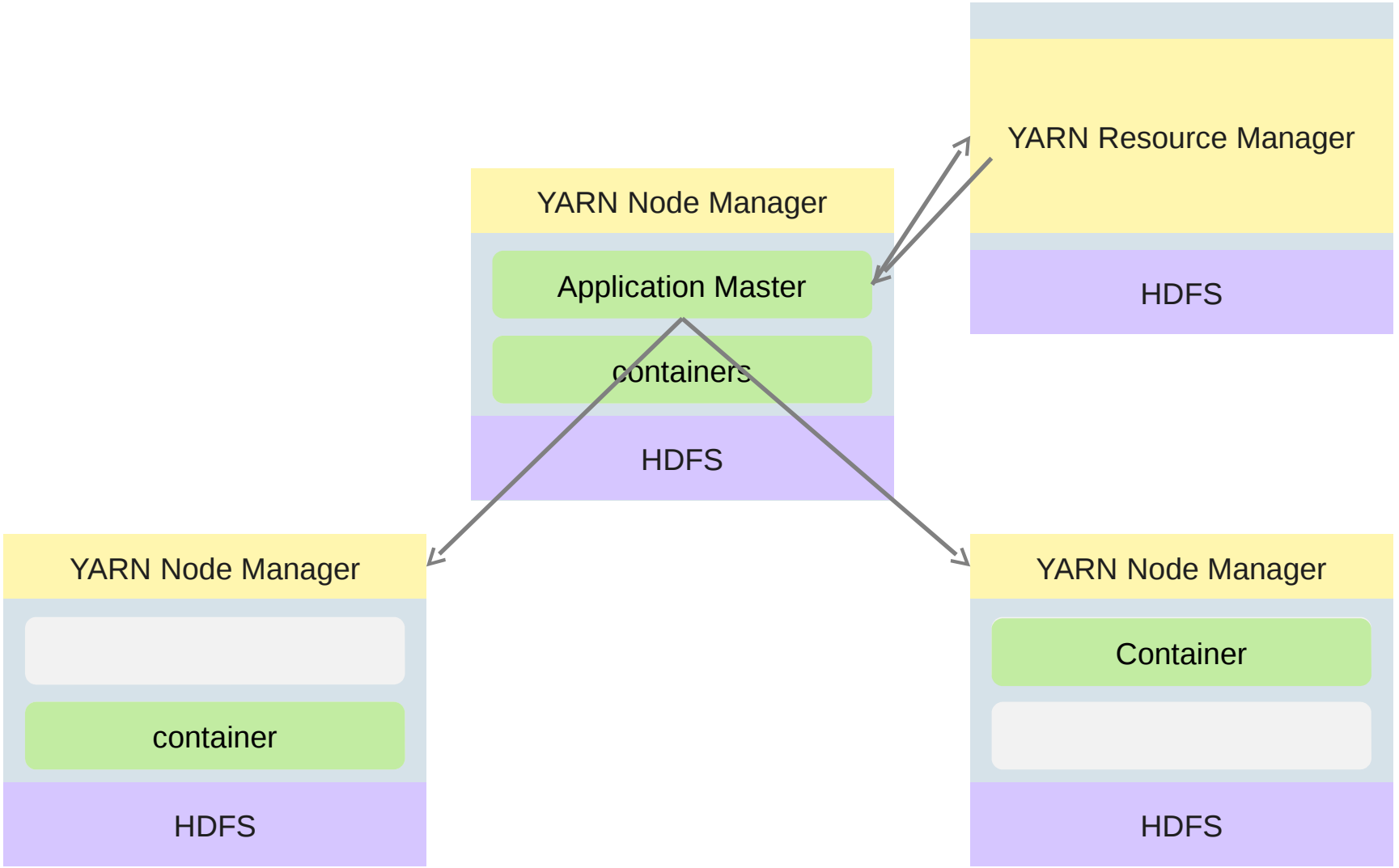
Container Requests

```
public static class ContainerRequest {
  final Resource capability;
  final List<String> nodes;
  final List<String> racks;
  final Priority priority;
  final boolean relaxLocality;
 ...
}
```

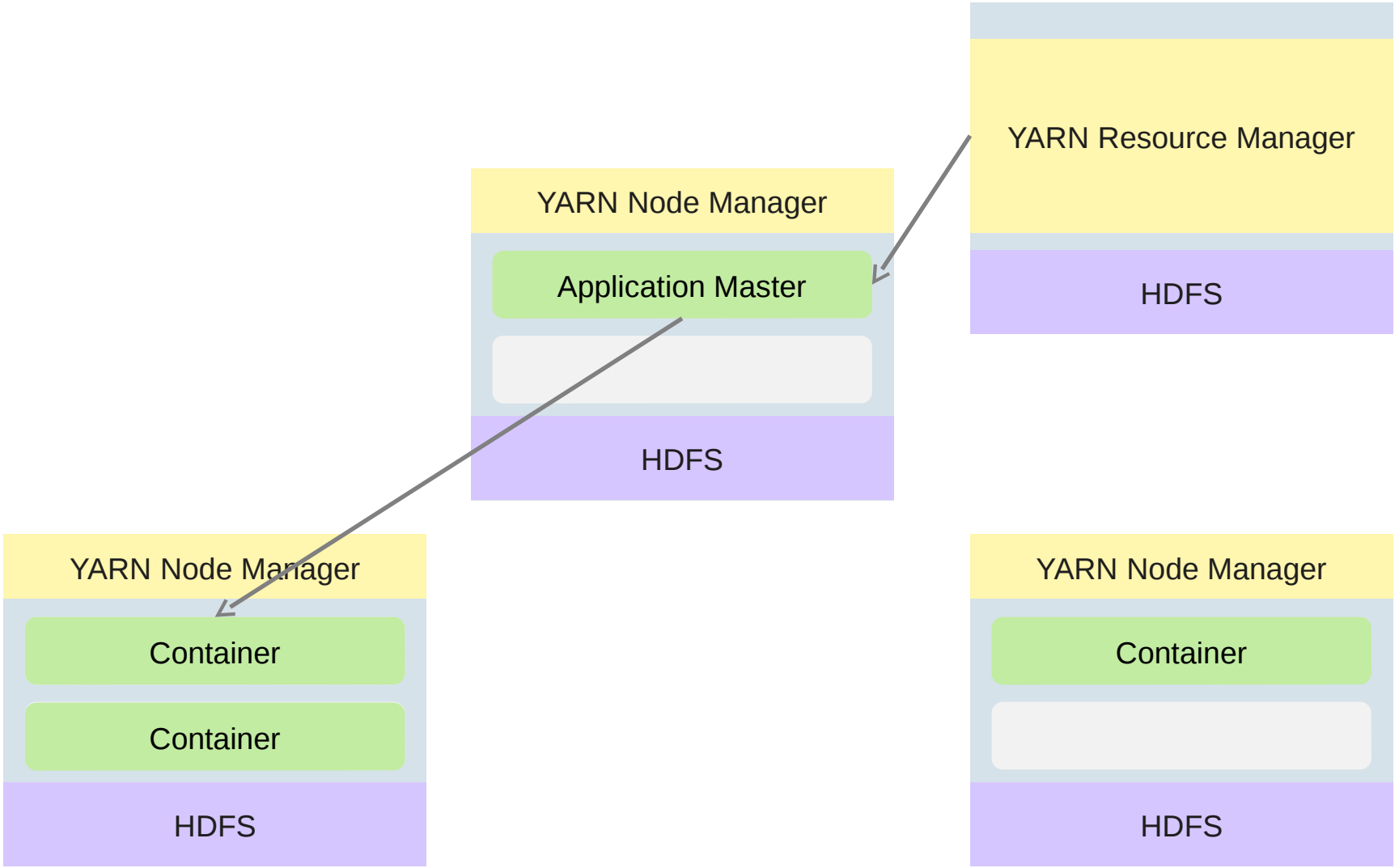
In Slider

- best-effort, persistent placement history
- some failure tracking
- TODO: moving average, *greylisting*

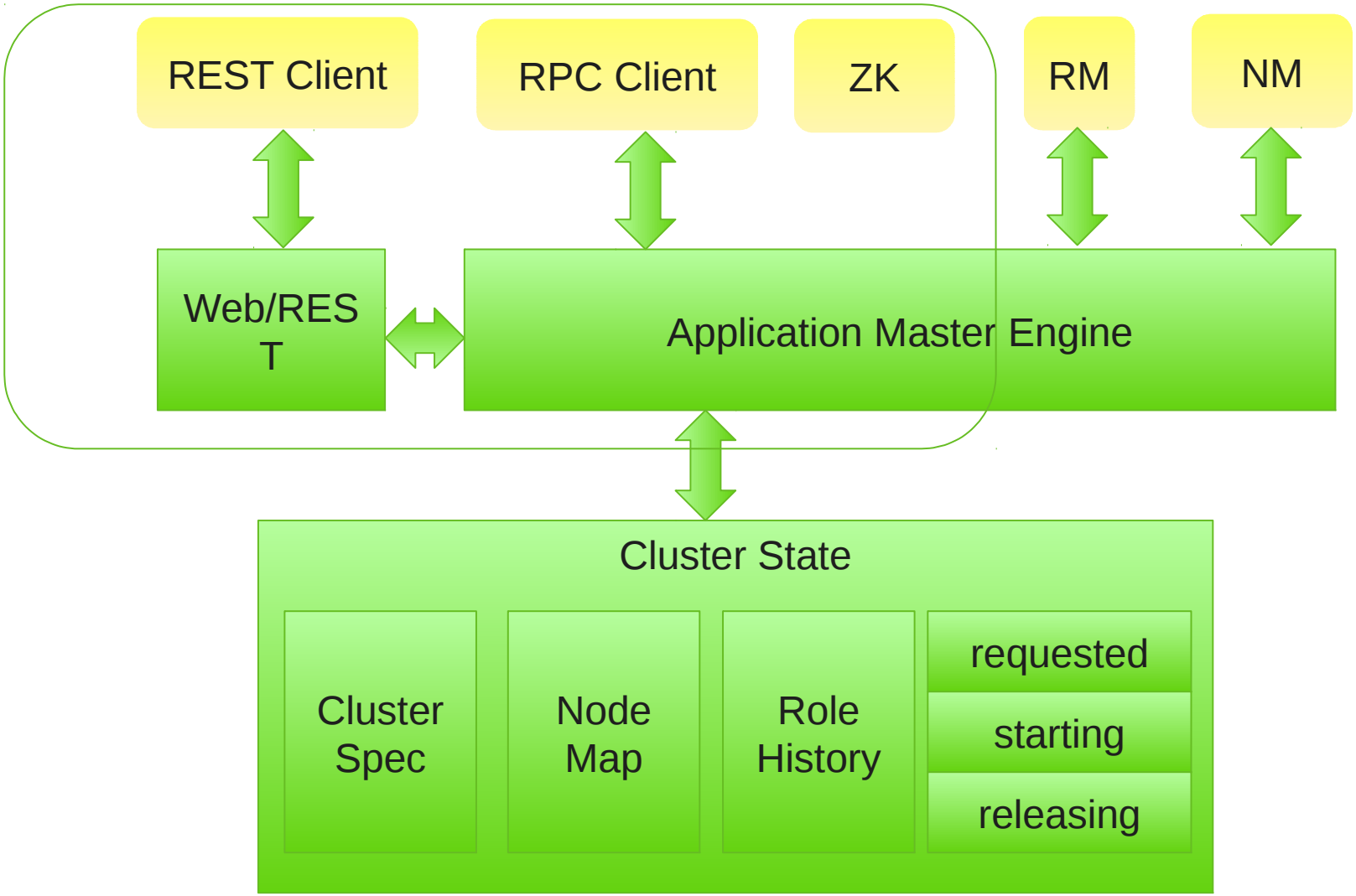
AM asks for containers



YARN notifies AM of failures



Remember Model-View-Controller



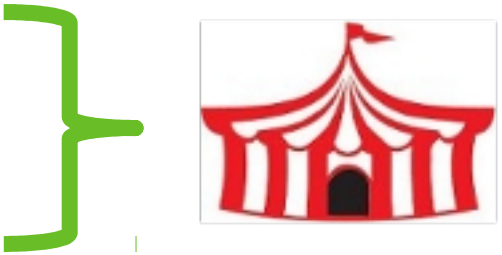
AM Restart –leading edge

Persisted in HDFS	Rebuilt	Transient
Specification resources.json &c	**NodeMap** model of YARN cluster	**Event History** application history
ComponentHistory persistent history of component placements	**Component Map** container ID -> component instance	**Container Queues** requested, starting, releasing

```
ctx.setKeepContainersAcrossApplicationAttempts(true)
```

Testing is fun

1. Unit tests
2. *?? Unmanaged AM??*
3. MiniYARNCluster
4. 1-node VMs
5. Cloud-hosted Hadoop clusters
6. Physical clusters



Me: Unit, Mini, Physical, VMs:

- RHEL 6 + Hadoop 2.4,
- Ubuntu 12, Java 8, Hadoop branch-2, Kerberos
- Windows 7, Hadoop 2.4

Avoid the Lamport Layer: delegate

- Spring XD: YARN apps in Spring
- Apache Tez: pipeline of operations, "sessions"
- Apache Slider : existing apps in a YARN cluster
- Apache Twill
- Microsoft Reef?

Twill: Runnables -> "elsewhere"

```
TwillController controller;

TwillRunnerService twillRunner = new YarnTwillRunnerService(
    conf, zkStr, createLocationFactory());
twillRunner.startAndWait();

TwillController controller = twillRunner.prepare(
  new RenderRunnable())
    .addLogHandler(new Slf4jLogHandler(log))
    .start();

Services.getCompletionFuture(controller).get();
```

Example: Frame Renderer

- render args + source image frame
- Generates GB of data
- Each frame render independent, repeatable
- Output-driven placement: consecutive frames “close”

github.com/steveloughran/renderize

actions from Twill context; IO to HDFS

```
public void run(){
 String[]aa = getContext().getApplicationArgum ents();
 RenderArgs args = new RenderArgs(aa);
 HadoopIm ageIO im ageIO = new HadoopIm ageIO(conf);
 BufferedIm age jpeg = im ageIO.readJPEG(args.im age);
 Renderer renderer= new Renderer(jpeg);
 int width = jpeg.getW idth();
 int height= jpeg.getHeight();
 int x = args.getRenderX(width);
 int y = args.getRenderY(height);
 renderer.render(x,y,args.m essage);
 im ageIO.w riteJPEG(renderer.im age,args.dest);
}
```

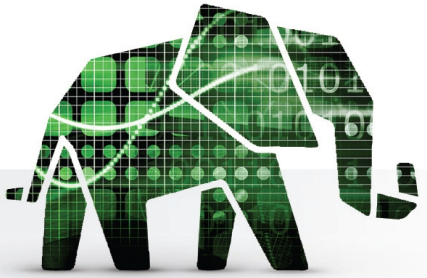
Summary : YARN

- Lets you run *whatever code you want* in a Hadoop Cluster
- Hides *a lot* of the tasks of deploying and running distributed apps
- Does require someone to handle the remainder
- Life is simpler if you can find someone else to do this: Twill, Spring XD, Tez, etc.
- …if you try, you can do lots of interesting things

Focus on the Algorithms

Questions?

hortonworks.com



Hortonworks

Security

Embrace Kerberos and test early

Set Security at launch time

```
if (insecureCluster) {
  env.put("HADOOP_USER_NAME",
    UserGroupInformation.getCurrentUser().getUserName());
} else {
  Credentials creds = new Credentials();
  String renewer = conf.get("yarn.resourcemanager.principal");
  hdfs.addDelegationTokens(renewer, creds);
  DataOutputBuffer dob = new DataOutputBuffer();
  creds.writeTokenStorageToStream(dob);
  ByteBuffer bytes = ByteBuffer.wrap(dob.getData(),
    0, dob.getLength());
  containerLaunchContext.setTokens(bytes);
}
```

And retrieve in AM

```
if (UserGroupInformation.isSecurityEnabled()) {
  secretManager.setMasterKey(
    response.getClientToAMTokenMasterKey().array());
  applicationACLs = response.getApplicationACLs();
}
```

Tokens expire after ~72 hours…

Hortonworks

IPC/RPC: Avoid

Hard to code, security painful,
Better: REST APIs
If you must do it: avoid protobuf

If you really, really must

```
META-INF/services/org.apache.hadoop.security.SecurityInfo:
   org.apache.slider.rpc.SliderRPCSecurityInfo

class SliderRPCSecurityInfo extends SecurityInfo { ... }

class SliderAMPolicyProvider extends PolicyProvider { ... }

@KerberosInfo(serverPrincipal= "slider.kerberos.principal")
public interface SliderClusterProtocol extends VersionedProtocol
 { ... }

//and in AM

rpcService.refreshServiceAcl(conf, new SliderAMPolicyProvider())
```

REST APIs

```
SliderAmIpFilter.doFilter(request, response, filterChain)
String dest = httpReq.getRequestURI();
if (!dest.startsWith("ws/")
  && !(proxyAddrs.contains(httpReq.getRemoteAddr()))) {
 String redirect =
   httpResp.encodeRedirectURL(proxy + dest);
 httpResp.sendRedirect(redirect);
 return;
}
```

Even if clients handled 307 response

Proxy doesn't forward PUT/POST/DELETE