Application environments are increasingly dynamic. For example, in AWS or Rackspace hosted environments, an administrator can bring up or take down nodes quickly due to workload or other factors. The result is an environment where there can be many, short-lived JVMs.

Considerations for a Dynamic Environment

The AppDynamics app agent operates substantially the same way in a dynamic environment as it does in a traditional data center. However, a few considerations apply when deploying agents in these environments:

- Ensure that the agents can reach your Controller. Particularly if your Controller operates on-premises while the agents run on hosted servers, you need to ensure that the agents can access the Controller through the firewalls in your environment.
- Configure custom node expiration handling. By default, the Controller manages the lifecycle for a node based on default timeout settings. You will likely want to reduce the default expiration times for nodes, as described in Historical and Disconnected Nodes. To manage this directly, rather than relying on activity timeouts, a strategy would be to run a script in the JVM before it shuts down that invokes the mark-nodes-historical resource in the AppDynamics REST API to declare itself historical to the Controller.
- Depending on the nature of your environment, it may not make sense to track each JVM instance as a distinct node in AppDynamics, since the set of nodes in this case would be boundless. Also, since the JVMs are identical, representing them as different nodes in AppDynamics does not best reflect the logical model of the environment. The following section provides more information.

Configuring the Agent for Dynamic Environments

You can configure AppDynamics to reuse node names for these types of environments. With node name reuse enabled, after a node is shut down and marked as historical in AppDynamics, the name of the node becomes available for reuse in subsequently registered nodes. You can also specify a node name prefix when reusing node names to indicate the association between all concurrently running JVMs for the dynamic application.

To configure the agent for this environment, use the following properties:

- **Reuse Node Name Property** to enable re-use of node names, and
- **Auto Node Name Prefix Property** to set the prefix used for automatically-named nodes

When enabled, the Controller manages name assignments for the agents. It forms node names by appending a sequentially incremented number as the suffix to the prefix you specify. Be sure to avoid specifying the node name using any other mechanism (such as in controller-info.xml file or via a system property).

The following listing shows sample settings for the properties as they may appear in a startup script:
When you have enabled reuse node name, the Java Agent logs to standard output after starting up and before registration with the Controller. To troubleshoot agent registration failure, you may want the log files to be written to a file instead.

To configure the agent to write logs to a file, edit the log4j-unknown.xml at `<agent_home>/<version_number>/conf/logging` and uncomment the AgentLog Appender, as follows:

```xml
<!-- To log files to an "unknown" folder before the agent registers with the Controller, uncomment the line below -->
<appender-ref ref="AgentLogAppender"/>
```

This change causes the log files of an agent prior to registration to be written to a directory named "unknown" rather than standard output.

### Marking Dynamic Nodes as Historical at Shutdown

As indicated in the [Configuring the Agent for Dynamics Environments](#) section, setting the node name reuse flag causes the JVM to report graceful shutdowns, allowing the controller to mark the corresponding node as historical. If you are not reusing node names, you can use one of the following alternative methods to have a node marked as historical:

- Set the following system property at JVM startup:

  ```bash
  -Dappdynamics.jvm.shutdown.mark.node.as.historical=true
  ```

- Set the following property in `app-agent-config.xml` to `true`, as shown:
<app-agent-configuration>
  
  <configuration-properties>
  
  <property
  name="appdynamics.jvm.shutdown.mark.node.as.historical"
  value="true"/>

  
</configuration-properties>
</app-agent-configuration>

**Note**

If your application environment abortively terminates JVMs rather than allowing them to shut down gracefully, arrange for your orchestration framework to use the controller API (Configuration API > Mark Nodes as Historical) to mark the node as historical at the same time as it terminates the node.

---

**Controller-side Handling of Historical Nodes**

If a node has been out of contact with the Controller for a certain amount of time, the Controller marks the node as a historical node. The Controller suspends certain types of processing activities for the node, such as rule evaluation. For more information, see *Historical and Disconnected Nodes.*