Cody's Outline

Thursday, October 24, 2019

2:12 PM

* 1. Cloud and Infrastructure
     1. Openshift --> Properties captured
        1. Automatically Detects:
           1. K8s pod name - user-provided name of the pod the container belongs to
           2. K8s container - name of container that runs the process
           3. K8s Full pod name - full name of pod the container belongs to
           4. K8s namespace - namespace to which the containerized process is assigned
           5. K8s pod UID - Unique ID of the related pod
        2. Can also search K8s labels and can use these tags for alerting profiles and filters

* 1. VMWare integration and metrics
     1. ESXi Host Health:
        1. CPU
           1. Percentage of actively used CPU of the host
           2. Active CPU is approximately equal to the ratio of used CPU to available CPU
        2. Memory
           1. Includes:

Memory usage: percentage of total RAM used by processes

Memory compression rate: memory compression saves memory but requires additional CPU cycles

Memory swapping: Rate at which memory is swapped between disk and active memory

* + - 1. Disk (storage I/O)
         1. Includes:

Throughput: Total number of bytes read and written to the disk per second

IOPS: I/O operations per second

Read/write latency: Time from I/O request submission to I/O request completion. This is the average delay of disk read/write operatins in milliseconds

Command queue latency: the average time spent in VMKernel queue per SCSI command

Aborted commands: The number of aborted SCSI commands. This value serves as the basis for ESXi host storage overload incidents

Number of datastores: Number of datastores connected to your vCenter. Includes metrics for free and used space.

* + - 1. NIC (network)
         1. Includes:

Traffic: Average transmitted/received network traffic throughput

Dropped Packets: Number of received/transmitted packets dropped during collection interval

* + 1. Integration
       1. VMware 
          platform 
          Hosts 
          2 
          Performance 
          data is sent to 
          ActiveGate 
          1 
          ActiveGate 
          retrieves 
          virtualization data 
          from VMware 
          ActiveGate 
          3 
          Data from vCenter 
          or ESXi host is sent 
          to Dynatrace for 
          analysis 
          OneAgent 
          continuously sends 
          host performance 
          data to Dynatrace 
          Dynatrace 

* 1. ActiveGate polls the VMWare platform (vCenter or standalone ESXi hosts) to obtain information about all important resources that an ESXi server provisions to your virtual machines. We also collect information about events such as virtual machine migrations and newly created machines.
  2. OneAgent automatically captures performance information related to CPU, RAM storage, NICs and more.
  3. By correlating information obtained from OneAgent and ActiveGate, we help you understand the dynamics of your virtual infrastructure and its influence on your server-side services and applications.

* 1. PCF and OneAgent
     1. OneAgent BOSH benefits:
        1. Foundation Health monitoring for facilitating platform and resource optimization
        2. Monitoring insights into all Cloud Foundry components, including Gorouters, Diego cells, Auctioneers, and more
        3. Health Metrics for each CF VM, including memory usage, CPU usage, Disk space usage, and Network IO
        4. Automatic monitoring of CF apps, down to the code and query level, thanks to built-in auto-injection for Garden-runC Linux and Winc Windows Server containers
        5. Automatic distributed service tracing, which provides an overview of the health of your entire microservices architecture
  2. CloudWatch API- how it works and what we need
     1. AWS
        1. General:
           + DT makes Amazon API requests every 5 minutes.
           + Amazon charges $0.01 per 1,000 metrics requested from CloudWatch API
           + Can be used with or without EAG
        2. How to:
           + Create AWS monitoring policy

Go to IAM in Amazon Console --> Policies --> Create Policy --> select JSON and paste from given page

* + - * + Choose access method:

Key-based

Need to rotate keys periodically.

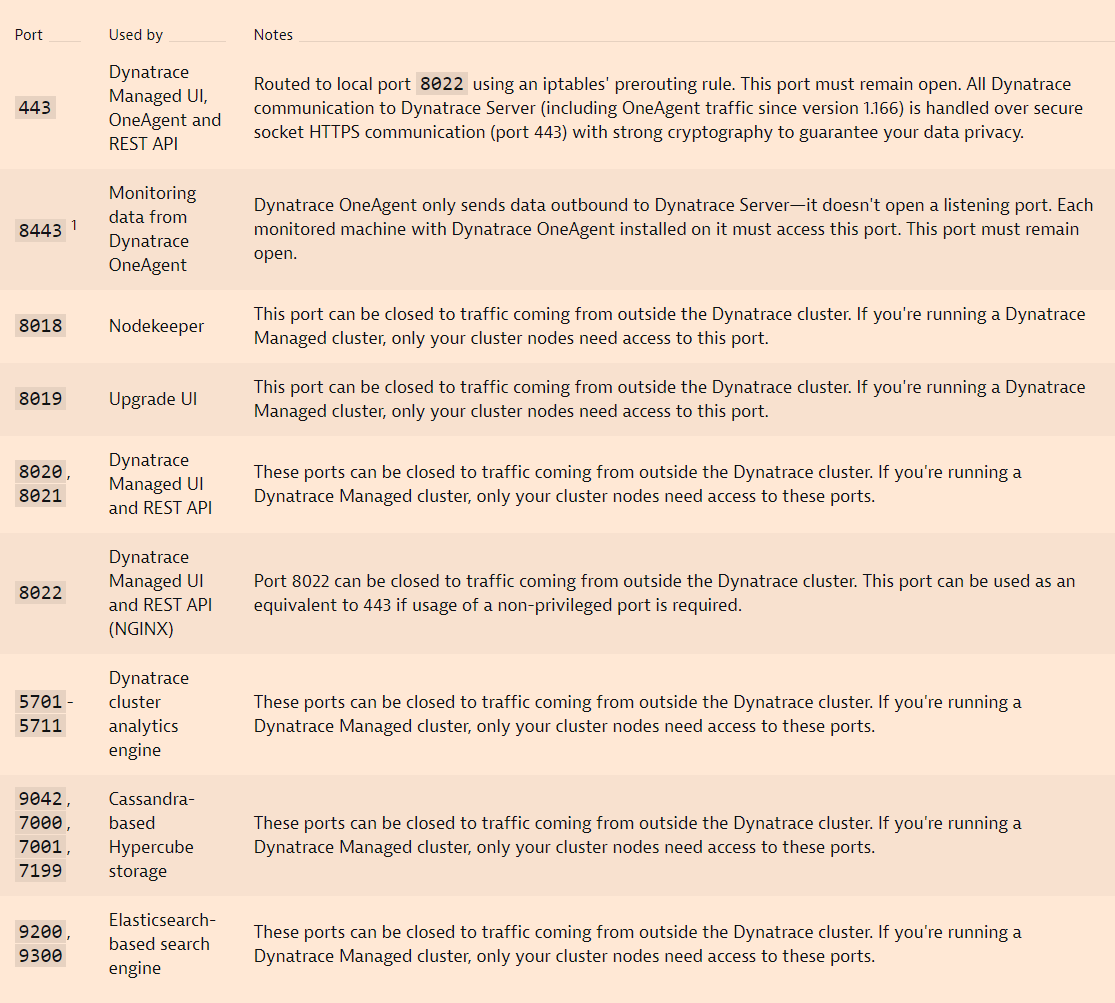
Needed to start: Rights to create new AWS user, AWS account ID, & Amazon Access Key ID & Secret access key

Role-based

Needed to start: AWS account ID, DT AWS Account ID (**509560245411),** Rights to assign role-based access to AWS account, External ID copied from Settings > Cloud & Virtualization >AWS

* + - * + Define AWS resource tagging

* 1. OpenStack capabilities- what we can monitor
     1. OneAgent monitors entire stack
     2. Auto-detects all virtualized components and keeps up with all changes
     3. Can be integrated with virtualized infrastructure
  2. Kubernetes vs OpenShift
  3. AWS CW monitoring and what we get
  4. OpenShift & Container Images
  5. EC2 Auto Scaling
  6. Docker and grouping processes
  7. Docker capabilities
  8. AWS CW tags
  9. OneAgent, Mobile App Monitoring, and Plugins
     1. Logs
        1. Can create s metric based on monitored logs. This metric can then be scanned for continuously within your logs and display a chart of that metric on your dashboard so any changes will be clearly visible. Must be based on processes or hosts.
        2. Can search for text patterns in log files to parse based on keywords, phrases, logical operators, and parentheses.
        3. Auto-discovery function plus custom log files can be used to add functionality.
        4. Can also add log files manually.
        5. Log files must match criteria to be detected.
        6. SaaS:
           + Log files are stored in Amazon Elastic File System in the zone where your DT environment resides.
        7. Managed:
           + Must provide Network File System to store log files centrally.
           + Mount path must be identical and available from all cluster nodes.
        8. Can mask sensitive data within your logs
        9. Masking rules are applied in order, i.e.: 1st masking rule is applied to the original log files, 2nd rule is applied to result of 1st rule, etc.
     2. OneAgent and Blocked Ports
        1. OneAgent consists of different processes that communicate via a TCP port with a watchdog. At startup, OneAgent watchdog 
           attempts to open the first available port between port 5øeee and *lee . In some cases you may need this port for your 
           own applications that are started after OneAgent. In such cases, you can change the port range that the OneAgent 
           watchdog uses by modifying the file watchdoguserconfig.conf . 
           The file watchdoguserconfig. conf is located in the following directory: 
           \ agent \ c onfig 
           You can change the watchdog listening port by modifying the following line in the file: 
           From, for example: -portrange=5eeee: 5910 
           To: 
           Be sure to restart OneAgent service following your changes. 
           Please see Which network U2LL.dQ±-Qyn.u.ac-S2L.Y.e.LU.x.Z for information on the ports used by Dynatrace. 

* 1. 
  2. JMX
     1. Java Management Extensions
        1. Can monitor any metric in your JVM that is exposed via an Mbean
           + Mbean is a managed Java object- follows the design patterns set forth in the JMX specification
           + Can represent a device, application or any resource that needs to be managed.
     2. Notes:
        1. Selecting a process/host/MZ does NOT limit the scope of the plugin monitoring. JMX/PMI plugins monitor all Java processes accessible to the OneAgent.
        2. Can turn off monitoring globally to determine on a host level which processes you want to monitor (consumes custom metrics)
  3. Process monitoring and exclusion
     1. Can specify which processes you want to monitor under Settings --> Processes & Containers --> Process Group Monitoring
     2. Built-in process monitoring rules:
        1. .NET and Go Kubernetes apps
        2. .NET and Go Cloud Foundry apps
        3. .NET and Go apps deployed in Docker containers
        4. ASP.NET Core applications started by IIS
        5. Core components of CloudFoundry written in Go
        6. Caddy - Web server written in Go
        7. InfluxDB - timeseries database written in Go
     3. Deep monitoring rules are only effective when OneAgent is installed on your hosts or images
     4. App-only integrations without a full OneAgent installation don't support monitoring rules.
  4. OneAgentMon
     1. Device that appears during Windows installation. Used for deep process monitoring. It works like a monitoring driver and allows DT OneAgent to add its own library between the OS and the processes it's monitoring.
  5. Custom Plugins and Workflow
     1. Can be created for any process that exposes an interface, such as processes that are served over HTTP (ex: DBs, apps, load balancers).
     2. To begin- use Python code and write a JSON file that describes your metrics and how you want to display them.
     3. OneAgent running the plugins can work in either full-stack or cloud-infrastructure monitoring mode.
     4. Can download DT plugin SDK for development
     5. Once the plugin is finished, upload to your environment
     6. Custom metrics will be shown under Further Details tab.
  6. OneAgent and Root
     1. Requires root privileges on Linux for:
        1. Installing OneAgent components in system library directories
        2. Setting up /etc/ld.so.preload to automatically monitor processes
        3. Adapting SELinux policies to allow for the monitoring of processes
        4. If Log monitoring is enabled:
           + Creating the DT log monitoring OneAgent configuration file
        5. Access list of open sockets for each process
        6. Access list of libraries loaded for each process
        7. Access name and path of the executable file for each process
        8. Access command line parameters for each process
        9. Monitor network traffic
        10. Parse executables for Go Discovery
        11. Gather monitoring data related to Docker containers
  7. [Agent install parameters](https://www.dynatrace.com/support/help/shortlink/windows-custom-installation)
     1. Windows group policy installer can be used with command line parameters when you can't use default settings.
     2. Can specify the following:

|  |  |
| --- | --- |
| **Term** | **Definition** |
| SERVER | Address of your DT Server, IP address or name, add a colon with port # following the IP |
| TENANT | Dynatrace Environment ID |
| TENANT\_TOKEN | Internal token that is used for authentication when OneAgent connects to the Dynatrace cluster to send data |
| HOST\_GROUP | Name of a group you want to assign the host to |
| PROXY | Address of the proxy server - use IP address or a name |
| APP\_LOG\_CONTENT\_ACCESS | When set to true, allows DT OneAgent to access log files for the purpose of log monitoring (possible values: true, false, 1, 0) |
| INSTALL\_PATH | Allows installation to a directory of your choice. Works only with absolute Windows paths |
| INFRA\_ONLY | Activates cloud infrastructure monitoring mode (possible values: 0 (deactivated) 1 (activated)) |
| USER | User used to run the process responsible for Dynatrace OneAgent plugin |
|  | Dtuser: default user account used to run DT components |
|  | No\_create: Disables user creation when installing DT OneAgent. Use this for deploying DT on Windows Server Domain Controller to avoid propagation of dtuser across the domain. **Using USER=no\_create disables DT OneAgent plugins functionality** |

* 1. AIX OneAgent
     1. Requirements:
        1. Admin rights for servers and changing firewall settings
        2. Permissions and credentials for restarting app services
        3. Host needs at least 300 MB RAM
        4. Not supported on networked storage mount points (NFS or iSCSI)
        5. All hosts that are monitored need to be able to send data to the DT cluster
           + Can be accessed directly or done using an ActiveGate
        6. Can inject manually if needed
           + IBM Java or IBM/Apache HTTP Server
  2. iOS instrumentation possibilities
     1. Can be instrumented via CocoaPods or Carthage-based approach.
     2. Manual Setup:
        1. Applications --> [Application] --> … --> Edit --> Instrumentation --> Apple iOS --> Developer and follow instructions
     3. Auto-instrumentation Features:
        1. Automatic OneAgent startup
        2. Lifecycle data
        3. Crash reporting
        4. Web request tagging
        5. Web request events execution time and the data sent are reported, 3-rd party requests are identified
        6. Web views
        7. Automatic user action detection
     4. These can be disabled/refined by adding configuration to Info.plist file.
     5. Configuration Keys: properties you can set to your preferences for auto-instrumentation. Add key's to Info.plist file

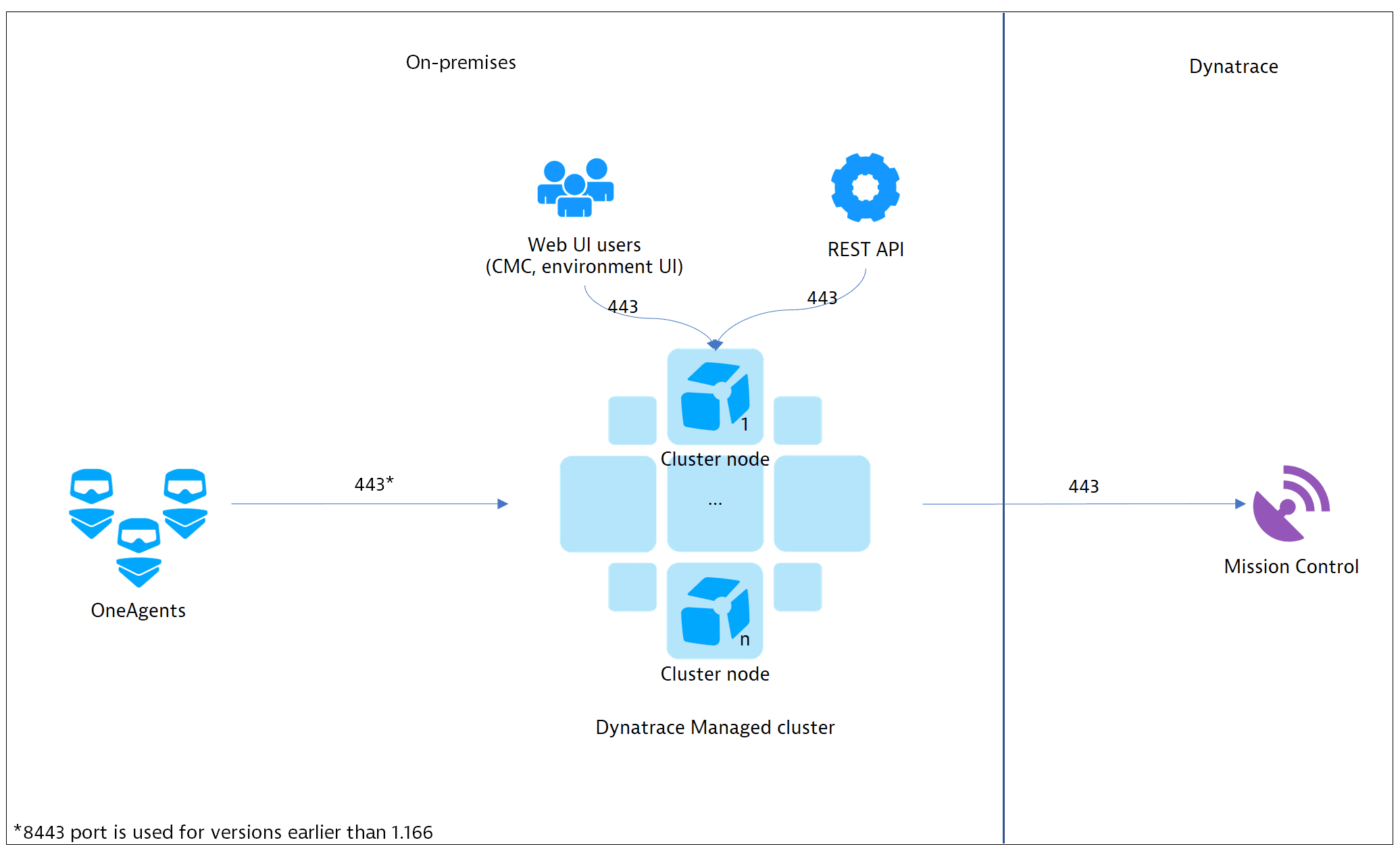
|  |  |  |
| --- | --- | --- |
| **Key** | **Key Type** | **Description** |
| DTXLogLevel | String | Values: ALL, FINEST, FINER, FINE, CONFIG, INFO, WARNING, SEVERE, OFF, it automatically enables OneAgent logging with set key. No value- automatically turned off |
| DTXApplicationID | String | Identifies mobile app. Error if not present. |
| DTXBeaconURL | String | Key's value is used to ID your environment within Dynatrace. Error if not present |
| DTXAgentCertificatePath | String | Defines path to a self-signed certificate in the DER format. Additional anchor to validate HTTPS communication. Default value is null. |
| DTXInstrumentWebViewTiming | Boolean | Turns on automatic web request timing and tagging for requests passed into a WebView. Default value is true. |
| DTXHybridApplication | Boolean | Used for hybrid apps, necessary to share the same visit for user actions created by RUM JavaScript tag. Default value is false. |
| DTXSetCookiesForDomain | String | Used for hybrid apps that use the JS library, cookies must be set for each instrumented domain or server that the app communicates with. Can specify domains, hosts, or IP addresses |
| DTXExcludedControls | Array | Defines array of items where each item contains a type of view or control to exclude from automatic creation of user actions. |
| DTXEcludedControlClasses | Array | Array of items where each item contains the name of a UI control class (or sub-class) to exclude from automatic control instrumentation. |
| DTXEcludedLifecycleClasses | Array | Array of items that contains the name of a class to exclude from automatic lifecycle instrumentation. |
| DTXCrashReportingEnabled | Boolean | Enables crash reporting. Default value is true. |
| DTXInstrumentLifecycleMonitoring | Boolean | Enables automatic lifecycle detection without the need to override your view controller classes with OneAgent SDK for iOS lifecycle classes. Default value is true. |
| DTXInstrumentWebRequestTiming | Boolean | Turns on automatic web request timing and tagging. Default value is true. |
| DTXInstrumentAutoUserAction | Boolean | Turns on ability to automatically create user actions for user interactions with the app. Default value is true. |
| DTXAutoActionTimeoutMilliseconds | Number | Sets the value for how long a particular automatic user action is action. Default value is 500 ms. |
| DTXAutoActionMaxDurationMilliseconds | Number | Sets the amount of time to retain an automatic user action before deletion. Default value is 60000 ms (60 seconds) |
| DTXSendEmptyAutoAction | Boolean | Determines whether to send automatic user actions that don't contain any web requests or lifecycle actions. Default value is true. |
| DTXInstrumentGPSLocation | Boolean | Captures the location only if the app uses CLLocationManager and sends the captured location as a metric to the server. Default value is true. |
| DTXUserOptIn | Boolean | When set to true, activates privacy mode. User content needs to be queried and set. Default value is false. |
| DTXAutoStart | Boolean | Set to false: agent does not start automatically and must be started by either calling startupWithInfoPlistSettings to start with the config from Info.plost or startupWithConfig to start with passed confi dictionary. Default value is true. |

* 1. Android instrumentation possibilities
     1. Automatic
        1. Gradle Plugin
           + Inserts OneAgent start-up at the beginning of the application start procedure.
           + Cannot use Instant Run feature or whitespace in project path.
           + Insert a snippet into your code
     2. Manual
        1. Include OneAgent library as a dependency
        2. Recommended to start OneAgent in the onCreate method.
        3. Can use OneAgent SDK to enrich mobile user experience data
        4. Limits:
           + Can only use general API methods for web requests tagging and timing for 3rd party web request frameworks
           + Limited support for multi-process apps
           + Does not support Direct Boot mode - have to manually startup the agent once device is unlocked
  2. Mobile App Dev framework support
     1. Android 4.0.3-10 (API 15-29)
     2. iOS 8+
     3. Frameworks:
        1. AFNetworking 3.x
        2. Apache Cordova 3.x+
        3. Custom WebViews Supported
        4. DexGuard Supported
        5. PhoneGap 2.9.1+
        6. ProGuard Supported
        7. OkHttp version 3 and 4 (and OKHttp-based libraries like Retrofit 2)
        8. Xamarin Xamarin.iOS, Xamarin.Android, Xamarin.Forms (.NET Standard > 1.6)
     4. Gradle plugin: 8.x, with Android gradle version 3.1-3.5, and gradle version 5.0-6.0
  3. Plugins and anomaly detection
     1. **Plugins can only run on physical or virtual hosts, not containers.**

* 1. Hybrid app frameworks
     1. Apache Cordova - <https://www.dynatrace.com/support/help/shortlink/cordova>
     2. Xamarin - <https://www.dynatrace.com/support/help/shortlink/xamarin>
     3. AngularJS 1.0-1.7
     4. Sencha Touch 2.0-2.4

* 1. Saas & Managed general and Others
     1. Event Log - paraphrased from here: <https://dev-wiki.dynatrace.org/display/ruxit/Dynatrace+Managed+-+Cluster+event+notification+matrix>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Issue** | **Severity** | **Comment** | **Event Message** | **Email sent?** |
| Process Crash (ES/Cass/SG/Server);  Watchdog restart not working/possible | 1st: Warning  2nd: Severe  Severe on single node and 2 cluster node  Warning on 3+ if 2+ nodes okay | Processes cannot be started via debugui, only restarted. | "Could not detect elasticsearch activity since [DATE]" | No |
| Node Down | Medium | Node unavailable & not reachable requires customer interaction | "Host <node id> is down" | Yes, if not in upgrade process |
| Network issues, between nodes | High if longer outages |  |  | No |
| Network issues to MC | Low on intermittent hiccups, high if longer outages | If a cluster does not send a heartbeat anymore, we cannot necessarily tell if its working or not. | "There is a lack of connection to Mission Control" | Yes |
| Disk space issues | High if data loss | Environment issue needs to be fixed by customer | "Not enough disk space on <mountPoint>" | No |
| Cluster upgrade issue, upgrade not started | Low | If "controlled" upgrade failed, we should know the root cause and how to fix | "Upgrade problem" | No |
| Cluster upgrade issue, not reconnected to MC | High | Needs support to interact with customer | Same as network issues to MC | No |
| Upgrade download issues | Low | Usually caused by network/connectivity issues | "<version> download failed on server <node id>" | No |
| Cluster issues: exception in logs of server, ES, Cassandra | Low to high |  |  | No |
| Unexpected shutdown events w/ recovery | Low | Tricky to detect |  | No |
| Agent/SGW issues | Low to high | Proactive silent fixes should be done if possible |  | No |
| Insufficient hardware | Low to high |  | "Insufficient hardware on server <node id>" | Yes |
| Transaction storage | Severe |  | "Transaction storage retention period has been truncated" |  |

* 1. Auditing and Compliance
  2. Cluster Management
  3. Installation of Managed
     1. **Scenarios:**
     2. Arrow indicates which component initiates communication
        1. **Basic: (internal only)**
           + 
           + Without further config, cluster is accessible only internally and exposes port 443 for REST API, OneAgent Traffic, and Web UI access. (CMC and environment UI)
           + By default, Remote access to MC is enabled
           + Each communication channel is secured with TLS
        2. **Pure Dynatrace Managed**
           + To receive monitoring data from external OneAgents or to use DEM monitoring services, you will need to expose the cluster to external networks and configure a public IP address
           + DEM services include:

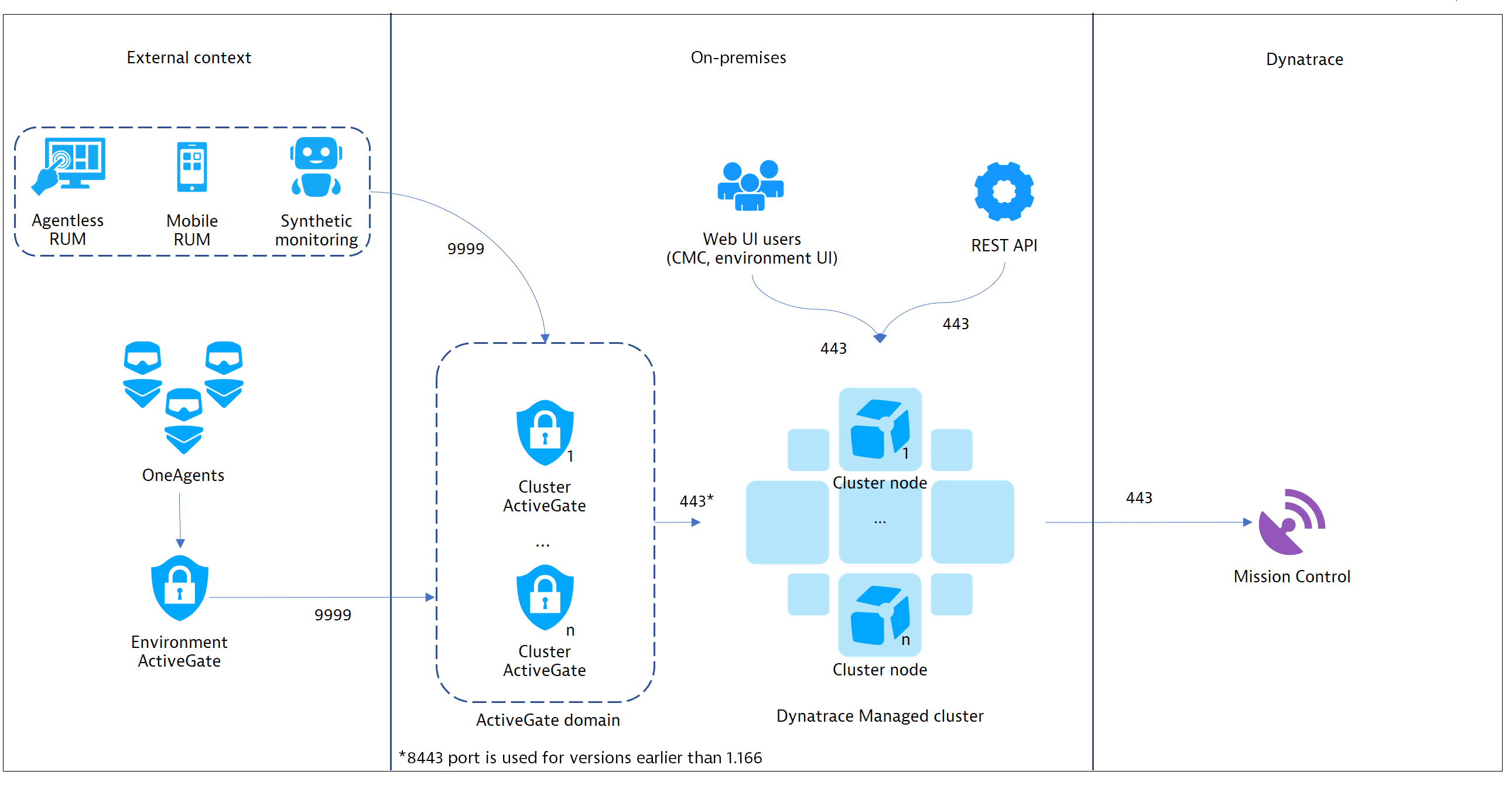
Synthetic monitoring

Agentless RUM

Mobile RUM

RUM via browser extension

Communication within the Dynatrace mobile app

* + - * + Exposing the cluster directly is not recommended, suggested to use 1+ Cluster ActiveGates as mediating proxies for pre-processing of OneAgent and DEM traffic.
        + CAGs will be recognized by cluster and can be configured through CMC
        + 
        + CAG requires:

Publicly available IP address

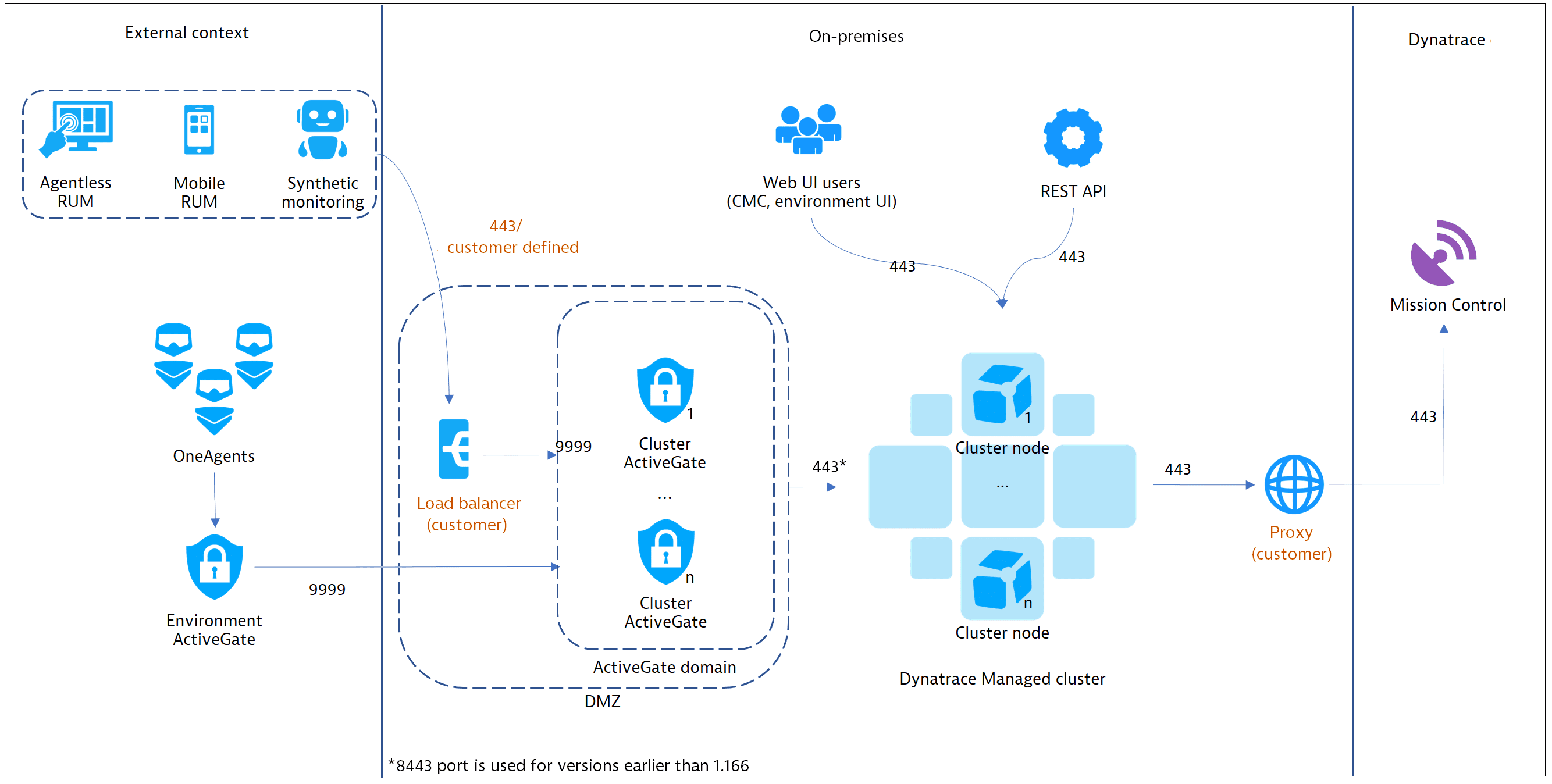
A domain name with a valid SSL certificate

* + - * + For high-load, production-ready installs with external hosts, apps, sessions, and synthetics, it's recommended to set up 2 load-balanced CAGs with the same domain name and certificate.
        + For smaller installs, can use a single CAG, or even install separate EAGs for each environment.
        + EAG starts communication with the CAG upon install, CAG must be operational and available via the public IP address beforehand.
        + **If using EAG:**

Set up DT Managed Cluster

Setup CAG and make sure it can connect to cluster and that CAG has a public IP address and is accessible outside network

Setup EAG and provide the CAG public address for external communication

* + - * + CAGs do not support proxying of Web UI traffic
      1. **Integration with existing IT landscape**
         1. Shows a customer-provided load balancer in front of the CAG domain and a customer-provided proxy for outbound communication to Mission Control
         2. ****
         3. Required Configuration for each traffic case

|  |  |  |
| --- | --- | --- |
| **Traffic type** | **Public IP** | **Valid SSL cert** |
| OneAgent (on premises) |  |  |
| OneAgent (external) | X |  |
| RUM (on premises) |  |  |
| RUM external |  |  |
| Agentless RUM (on premises) |  | X |
| Agentless RUM (external) | X | X |
| Mobile RUM (on premises) |  | X |
| Mobile RUM (external) | X | X |
| Synthetic | X |  |

* 1. Disconnect from MC
     1. How to?
     2. What to do if disconnected?
  2. Cluster Management API
     1. Available in CMC --> User Menu --> Cluster Management API
     2. Backup configuration 
        Cluster 
        Cluster node 
        Firewall configuration 
        Public endpoints 
        Remote access 
        SMTP settings 
        SSL certificates 
        SSO configuration 
        Startup configuration 
        Support Archives 
        Updates 
        User groups 
        I-Iser repository configuration 
        User session 
        Users 

* 1. MC principles and data sent
     1. Data sent:
        1. Usage and billing information
        2. Dynatrace Cluster health
           1. Such as number of nodes, status of Dynatrace services, or disk partitions usage
        3. Dynatrace cluster events
           1. Server starts/shutdowns, added/removed nodes, ActiveGate registrations
        4. Cluster settings
        5. Software Updates
     2. Security:
        1. Communication performed via HTTPS with browser-like certificate checks.
  2. Backups
  3. Ports
  4. Tokens
  5. API Throttling
  6. SaaS API
  7. Log Analytics basics
  8. Log analytics patterns
  9. DQL
  10. UEM
      1. **Conversion**
      2. **UEM Deployment**
      3. **How UEM works**
      4. User sessions
         1. Def: A group of user actions that are performed in your web application during a limited period of time. A single session typically includes: multiple page loads, 3rd-party content requests, service requests, and user actions. Each session includes at least one user action
         2. User session started when the first user action is initiated
         3. User session ends:
            1. After 30 minutes of inactivity
            2. When user closes their browser (not just when a tab is closed)
            3. Once the limit of 200 user actions per session is reached
            4. By calling JavaScript API function dtrum.endSession(). This can be used within your app to end user sessions automatically when (for example) a user logs out. Can also be used for testing.
         4. User sessions appear 2-4 minutes after the end of the session. Can sometimes exceed to 10.5 minutes.
         5. User session grouping:
            1. Some information is stored in session cookies and local storage, this enables the correlation of user actions into one user session. This session is erased when: Browser is closed, user clears their browser cookies, or user cleans up their local storage.
         6. User session duration: Elapsed time between initiation of the first user action in a session and the completion of the last user action in the session. Timeout periods aren't factored into user session duration measurements.
      5. Retention
         1. Data retention by type 
            Data type 
            Distrihuted trace and code 
            insights 
            Secy.ices;-ße.quests.-and-reques.t 
            (waterfall analysis, JavaScript errors, 
            and crashes) 
            RUM: User sessions 
            Synthetic 
            LogMonitQLing 
            Dynatrace SaaS 
            10 
            35 
            10 
            35 
            35 
            days 
            days 
            days 
            days 
            days 
            Configurable, with maximum 
            35 days of retention time 
            35 days 
            Configurable from 5-90 days. 
            Specific files can be 
            included/excluded. 
            Timeseries metrics (Key user actions and 
            Unlimited 
            requests) 
            Dynatrace Managed 
            Configurable, with maximum 
            365 days of retention time 
            Configurable, with maximum 
            365 days of retention time 
            Configurable, with maximum 
            35 days of retention time 
            Configurable, with maximum 
            35 days of retention time 
            35 days 
            Configurable, with maximum 
            35 days of retention time 
            Configurable, with maximum 
            35 days of retention time 
            Configurable from 5-90 days. 
            Specific files can be 
            included/excluded. 
            Unlimited 
            Storage 
            Proprietary; Shared with 
            non-aggregated RUM 
            data 
            Proprietary 
            Shared with distributed 
            trace and code service 
            insights 
            File-based NFS storage; 
            Storage-requirements 
            Cassandra 
         2. What they include:
            1. **Distributed trace and code insights**: PurePath data
            2. **Requests and Request Attributes**: 10 second granularity of charts, non-key and key requests
            3. **Non-aggregated user action data**: Full details of every user action for 10 days. This enables you to analyze individual user actions and get all details including waterfall analysis, JavaScript errors, and mobile crashes for 10 days.
            4. **Aggregated user action data:** Aggregated user action metric available for 35 days. After 10 days, user actions data is optimized for aggregated views and some individual user actions become unavailable for individual analysis. Sample set is large enough for statistical correct aggregations.
            5. **User Sessions**: Session Replay data. All user session data is stored for 35 days. Note that waterfall analysis, JavaScript error, and crash data are stored with RUM non-aggregated user action data.
            6. **Session Replay**: Min. size of required Session Replay storage volume is entirely load-dependent. A max size isn't required. In SaaS deployments, a dedicated disk is used for Session Replay data. In managed, Session Replay data storage is a dedicated file store that's used exclusively for Session Replay data.
            7. **Log Monitoring:** For SaaS: log files are stored in Amazon Elastic File System in the zone where your DT environment resides. Disk storage costs are included in your Log Monitoring subscription.

For Managed: Log files can be stored centrally once a common Network File System mount point is provided. It must be identical and available from all cluster nodes. With this, it is the user's responsibility to ensure appropriate levels of performance, availability, and free space on the mounted NFS volume.

* + - * 1. **Memory Dumps:** Immediately deleted from the disk once they're uploaded to the ActiveGate. When an upload isn't possible, memory dumps up to 20GB are stored on the disk for up to 2 hours.
        2. Timeseries Metrics:

0-14 Days: 1- minute interval granularity available for dashboarding and API access

14-28 Days: 5-minute interval " "

28-400 Days: 1-hour interval " "

400+ Days: 1-day interval " "

* + 1. **UEM Problem Analysis Workflow**

* 1. Anonymous vs Tagged Users
     1. Anonymous session: Non-authenticated users identified via browser cookies, or when a user tag has been edited or deleted.
     2. Web applications: Persistent cookie is used within each user's browser to identify a user.
     3. Mobile: Identified by specific mobile device they use.
     4. As long as a user has logged into the application at least once, you can search for and identify that user, even when user access is anonymous, unauthenticated sessions.
  2. Apdex
     1. Calculated for each discrete user action and each application overall. Provides insight to user experience
     2. Rating Scale:

|  |  |
| --- | --- |
| **Measurement** | **Rating** |
| 0.94-1.0 | Excellent |
| >.85 | Good |
| 0.7-0.85 | Fair |
| <0.7 | Poor |
| <0.5 | Unacceptable |

* 1. Errors and HTTP status codes
     1. User actions with JS errors are rated as Frustrated, even if the user actions are fast and below the Apdex threshold, but are colored red and indicated as Frustrated due to JS errors.
  2. Configuration
     1. Can adjust Apdex based on an individual application
  3. Entry Actions
     1. First page load or XHR actions that are typically the landing pages of an application.
     2. Indicate when a user initially uses the application
  4. Key User Actions
     1. Benefits:
        1. Can allow you to customize the Apdex of an individual action
        2. Can make a dashboard tile for the action
     2. Max of 20 key user actions per application
     3. To mark as Key User Action:
        1. Application --> Top 3 User Actions
        2. View Full Details --> Top 100 User Actions
        3. Click to the top right corner of an action and mark as key user action
  5. User Tagging
     1. Default: Assigns a unique, random ID to each new user. Can assign more meaningful user tags ( user names, email address, etc.)
     2. Based on Page Metadata:
        1. Can be used: DOM element text, meta tag, JS variable, cookie attribute
        2. Can be selected using CSS selector
     3. Based on Request Attribute:
        1. Application --> Edit --> User Tags --> Add tag (identifier) rule --> Expression type to capture --> Server side request attribute --> Request Attribute name
        2. Note that the service request attribute may not be captured for every user action/user session due to Adaptive Capture Control.
     4. **The last user action in a session that contains a tag will be used as the tag for the entire session**
     5. Can also identify mobile users of native mobile applications
  6. User Action naming rules
     1. Needed when certain session IDs are not automatically recognized and removed from certain user action names
     2. Action name detection
        1. Tries to assign meaningful names for actions, to do so it checks several action properties (inner HTML, caption, HTML element that triggers the actions).
        2. This element can either be a button or an anchor, it also tries to get the caption if there's a more complex HTML structure within multiple nested tags.
        3. **Data-dtname**: can set the custom attribute within HTML tags and use it as a caption
  7. Injection techniques
     1. Automatic:
        1. OneAgent JS tag injection is performed during installation for **Java, Apache HTTP Server, IIS, NGINX, and Node.js**. Just need to initiate injection and restart web server processes. OneAgent identifies HTML content responses and automatically injects a small JS tag into the head section of each page.
        2. JavaScript sends data back to your web server. OneAgent performs some initial monitoring of the data then forwards it to Dynatrace server. Transmission of data is performed using XHR Post requests.
     2. Manual:
        1. Can paste the tag into pages you intend to monitor. Can define a custom location for tag injection by adjusting application settings. For Java, you can specify injection within the innermost or outermost HttpServletResponse, within a custom servlet, or within a filter.
        2. Custom injection settings:
           1. Application --> Edit --> Advanced Setup --> JavaScript Injection
        3. Data is sent directly to Dynatrace, bypassing your infrastructure.
  8. Problems
     1. Problem Workflow
     2. Anomaly Detection Mechanism
     3. Baselines
        1. Baseline cube is calculated 2 hours after your application/service is initially detected by OneAgent so that it can analyze 2 hours of actual traffic to calculate preliminary reference values and identify where traffic comes from. Calculation of the reference cube is repeated every day so that Dynatrace can continue to adapt to changes in your traffic.
        2. Automated Multi-dimensional:
           1. Huge amount of high-quality and accurate data is necessary to determine baselines that can effectively be used to distinguish between normal and anomalous situations. Dynatrace AI learns the typical reference values of application and service response times, error rates, and traffic to reduce change of false positive.

**Traffic:**

Based on assumption that most business traffic follows predictable daily and weekly traffic patterns. Dynatrace automatically learns each applications' unique traffic patterns. Alerting on traffic spikes and drops begins after a one week learning period that is used to determine daily and weekly traffic patterns. It then uses this to forecast the next week's traffic and compares to the actual incoming application traffic with the prediction. If Dynatrace detects a deviation from forecasted traffic levels that falls outside of reasonable statistical variation, it raises an alert.

**Error Rate:**

Alerting on an error rate increase begins once the baseline cube is ready and the application/service has run for at least 20% of a week. Each baseline cube cell also contains the measured error rate. It also adapts to individual browser versions that can show either a higher or lower error rate compared to other browser types.

**Response Time:**

Dynatrace collects references for the median (above which are the slowest 50% of all callers) and the 90th percentile (the slowest 10% of all callers). A slowdown is raised if the typical response times for either the median or 90th percentile degrade.

Special emphasis is placed on the 10% of slowest response times experienced by your customers. This is because if you only know the average (median or mean) response times experienced by the majority of your customers, you'll miss a crucial point: Some of your customers are experiencing unacceptable performance problems.

Alerting based on response time degradation is also available once the baseline cube is ready and the application/service has run for at least 20% of a week.

* + - 1. Baselining Dimensions:
         1. Application:

User Action: An application's user action

Geolocation: Hierarchically organized list of geolocations where user sessions originate.

Browser: Hierarchically organized list of browser families, such as Firefox and Chrome.

Operating System: Hierarchically organized list of operating systems, such as Windows and Linux.

* + - * 1. Service:

Service Method: A service's individual service methods. In the case of database services, the service method represents the different SQL statements that are queried.

Service Method Group: Static or dynamic groups for web services, and for database services, groups that correspond to database operations like insert, update, select, and so forth. For database services, a reference value is calculated for the predefined service method groups inserts, updates, and selects.

* 1. Static Thresholds
     1. Infrastructure monitoring is based on numerous built-in, predefined static thresholds.
     2. Can be seen in Settings --> Anomaly Detection --> Infrastructure
     3. For applications and services you can disable automated baselining-based reference-value detection and switch to user-defined static thresholds.
     4. A slowdown event is raised if the static thresholds for either the median or the 90th percentile response times are breached.
     5. They cannot adjust to changing traffic patterns.
     6. Predefined Static Thresholds:
        1. Hosts:
           1. **CPU Saturation**: Alert if CPU usage is higher than 95% in 3 of 5 one-minute intervals
           2. **Memory usage event**: Alert if memory usage is higher than 90% on Windows or 80% on Linux AND memory page fault rate is higher than 100 faults/s on Windows or 20 faults/s on Linux in 3 of 5 one-minute intervals
           3. **GC activity:** Alert if GC time is higher than 40% OR GC suspension is higher than 25% in 3 of 5 one-minute intervals
           4. **Java out of memory:** Alert if the number of Java out-of-memory exceptions is 1 per minute or higher.
        2. Networks:
           1. **Number of dropped packets**: Alert if receive/transmit dropped packet percentage is higher than 10% AND total packets rate is higher than 10 packets/s in 3 of 5 one-minute intervals
           2. **Network Utilization**: Alert if sent/received traffic utilization is higher than 90% in 3 of 5 one-minute intervals
           3. **TCP connectivity for process**: Alert if percentage of new connection failures is higher than 3% AND number of failed connections is higher than 10 connections/min in 3 of 5 one-minute intervals
           4. **Retransmission Rate**: Alert if retransmission rate is higher than 10% AND number of retransmitted packets is higher than 10 packets/min in 3 of 5 one-minute intervals
        3. Disks:
           1. **Low disk space**: Alert if free disk space is lower than 3% in 3 of 5 one-minute intervals.
           2. **Slow running disks**: Alert if disk read time and write time is higher than 200 ms in 3 of 5 one-minute intervals.
           3. **Inodes number available:** Alert if percent of available inodes is lower than 5% in 3 of 5 one-minute intervals.
  2. Failure Analysis
  3. Anomaly Detection Learning Period
  4. Reoccurring Problems
  5. Alerting Profiles
  6. Problem Dimensions
  7. Deployment Changes
  8. Webhooks
  9. Problem API and feed
  10. Problem Severities
  11. Problems vs Events
      1. Problem: Represents anomalies in environment.
      2. Event: When a problem is detected and a problem event is created.
  12. Synthetics
      1. Synthetic and Applications
      2. Synthetic and Large Resources
      3. Synthetic Outage Handling
      4. Intervals and Locations
      5. Device Profiles
      6. Modifying a Test
      7. Authentication
      8. Visually Complete
  13. Things to Review in Detail
      1. Know EXACTLY what the managed installer does (OS level)
      2. Know what MUST be sent from a managed cluster to mission control
      3. Research OneAgent required permissions
      4. Review the Cordova plugin requirements and supported technology
      5. Review OpenShift in detail
         1. Ways to deploy
            1. **Full-Stack**

**Details**

Openshift environments that allow running privileged containers on nodes

OOTB, automated cluster and workload monitoring

One-shot, setup once, monitor everything with OneAgent

OneAgent auto-injection of code modules into pods for full-stack

Rolled out via Kubernetes-native vehicles like **OneAgent Operator or DaemonSet**

**Preferred for:**

OpenShift Container Platform Clusters

* + - * 1. **Application Only**

**Details**

For locked-down OpenShift environments without access to nodes

Workload monitoring on a per Docker image basis, no cluster nodes visibility

Dynatrace OneAgent code modules integrated with every Docker image

OneAgent universal-injection of code modules

Rolled out as part of normal Openshift workloads

**Preferred for:**

OpenShift Online

* 1. Metrics and information captured and displayed
     1. Actual CPU/Mem usage of cluster nodes (min, max, median)
     2. Total of CPU/Mem requests of containers running on cluster nodes (min, max, median)
     3. Total of CPU/Mem limits of containers running on cluster nodes (min, max, median) - limits may be overcommitted (over 100%)
     4. Available CPU/Mem resources for running additional pods/workloads on cluster nodes (min, max, median)
     5. Total CPU that can be allocated for pods (since part of the CPU is typically reserved by the system)
     6. Max memory that can be allocated for pods
  2. Review Docker in detail
     1. Permissions needed
     2. Deploying on host vs container
  3. Familiarize with all options for Synthetic Tests
     1. Outages and Thresholds