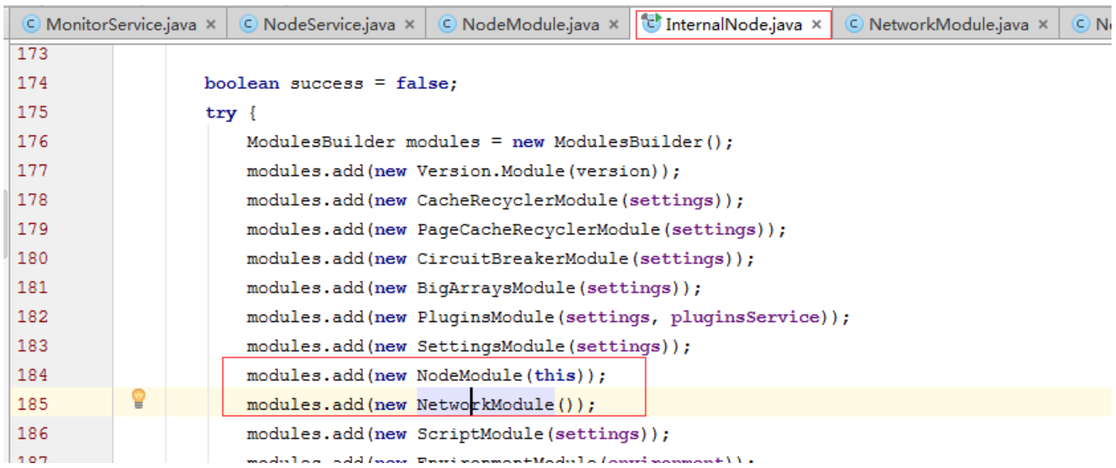
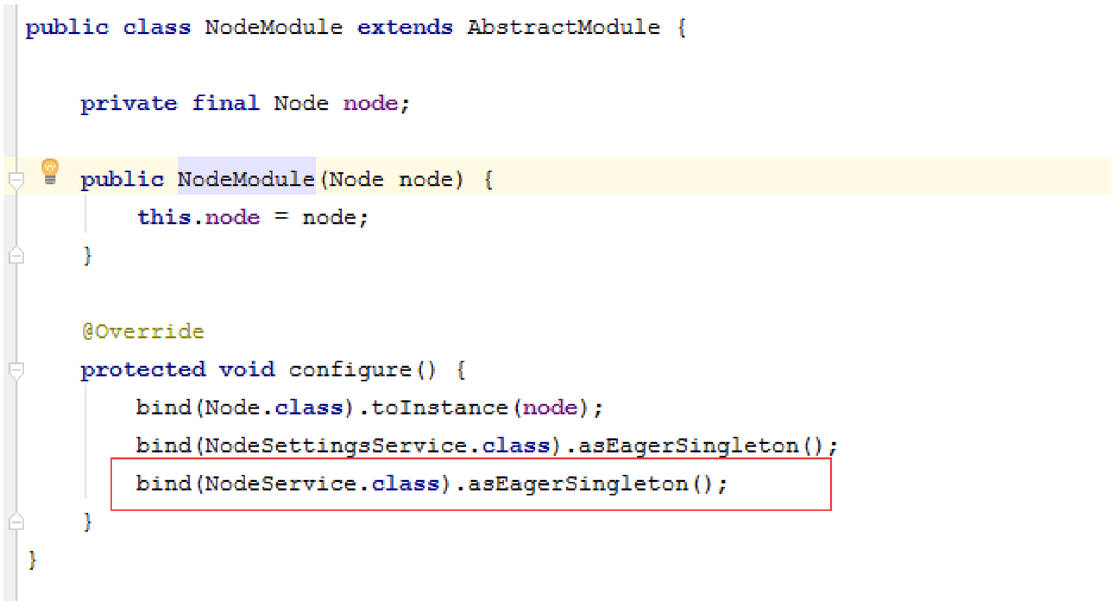
扩展阅读资料：
Java 系统运行时性能和可用性监控
https://www.ibm.com/developerworks/cn/java/j-rtm1/

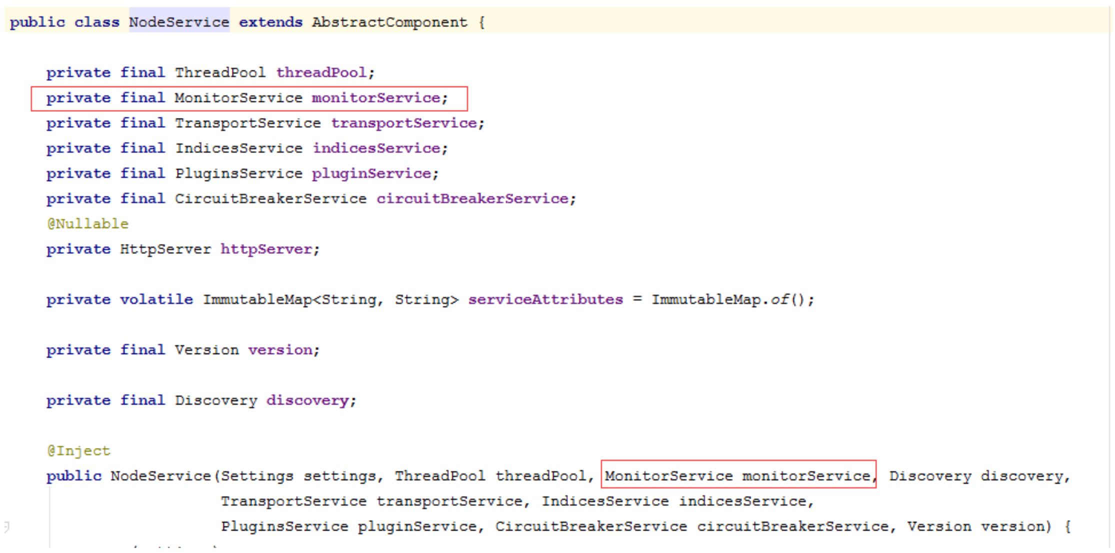
监控涉及的技术，监控采集的过程



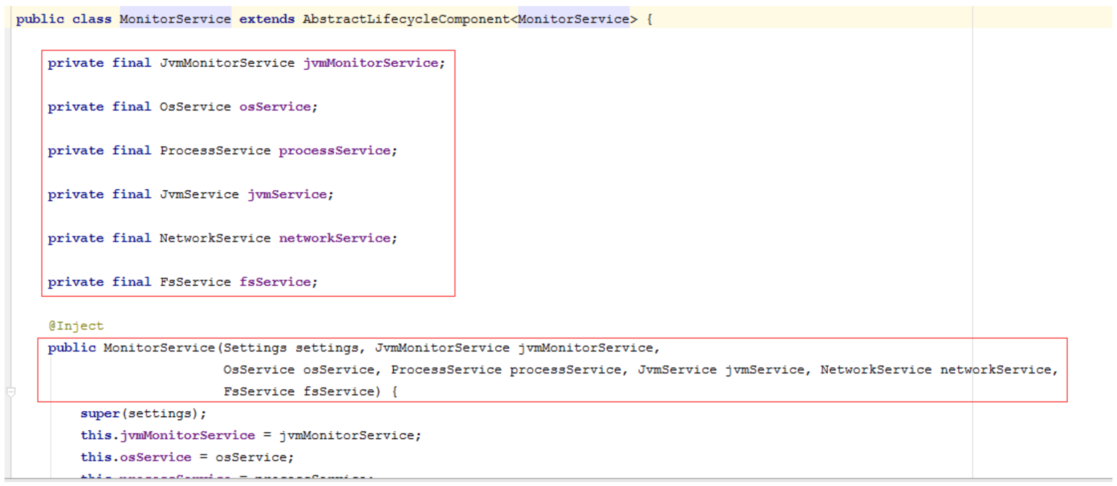
```java
        boolean success = false;
        try {
            ModulesBuilder modules = new ModulesBuilder();
            modules.add(new Version.Module(version));
            modules.add(new CacheRecyclerModule(settings));
            modules.add(new PageCacheRecyclerModule(settings));
            modules.add(new CircuitBreakerModule(settings));
            modules.add(new BigArraysModule(settings));
            modules.add(new PluginsModule(settings, pluginsService));
            modules.add(new SettingsModule(settings));
            modules.add(new NodeModule(this));
            modules.add(new NetworkModule());
            modules.add(new ScriptModule(settings));
```



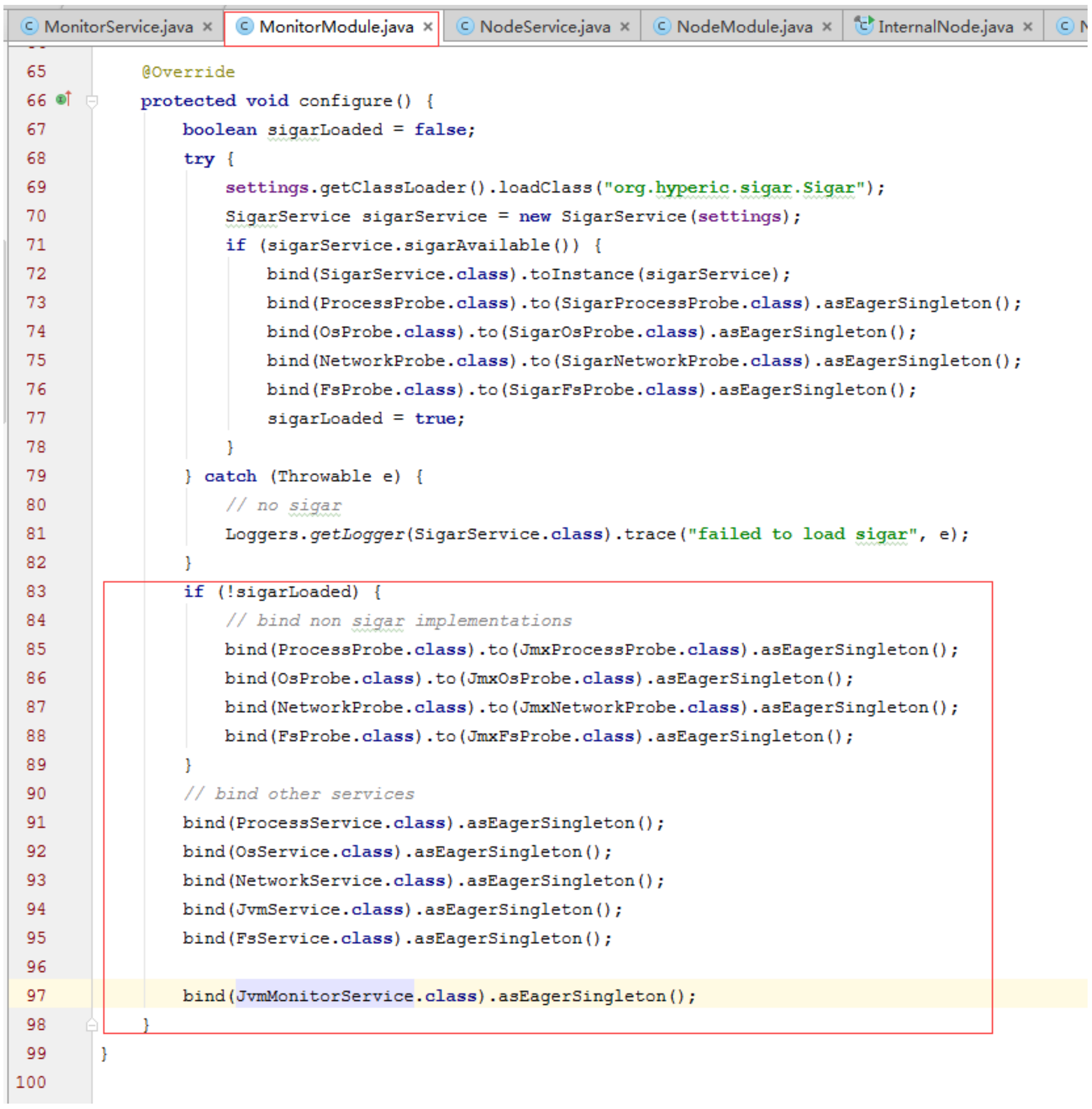
```java
public class NodeModule extends AbstractModule {

    private final Node node;

    public NodeModule(Node node) {
        this.node = node;
    }

    @Override
    protected void configure() {
        bind(Node.class).toInstance(node);
        bind(NodeSettingsService.class).asEagerSingleton();
        bind(NodeService.class).asEagerSingleton();
    }
}
```



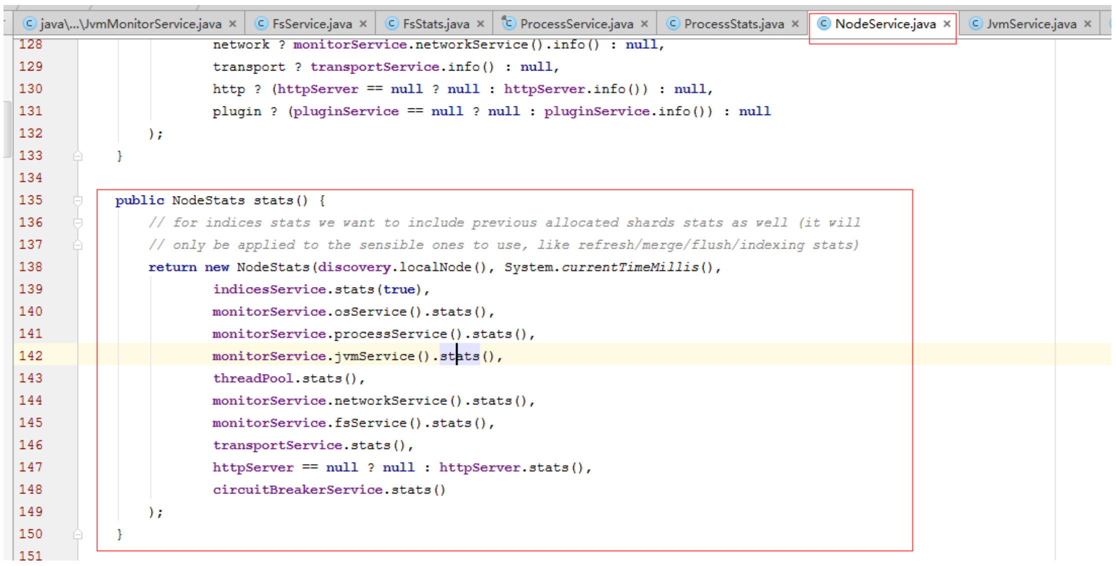
```java
public class NodeService extends AbstractComponent {

    private final ThreadPool threadPool;
    private final MonitorService monitorService;
    private final TransportService transportService;
    private final IndicesService indicesService;
    private final PluginsService pluginService;
    private final CircuitBreakerService circuitBreakerService;
    @Nullable
    private HttpServer httpServer;

    private volatile ImmutableMap<String, String> serviceAttributes = ImmutableMap.of();

    private final Version version;

    private final Discovery discovery;

    @Inject
    public NodeService(Settings settings, ThreadPool threadPool, MonitorService monitorService, Discovery discovery,
                       TransportService transportService, IndicesService indicesService,
                       PluginsService pluginService, CircuitBreakerService circuitBreakerService, Version version) {
```



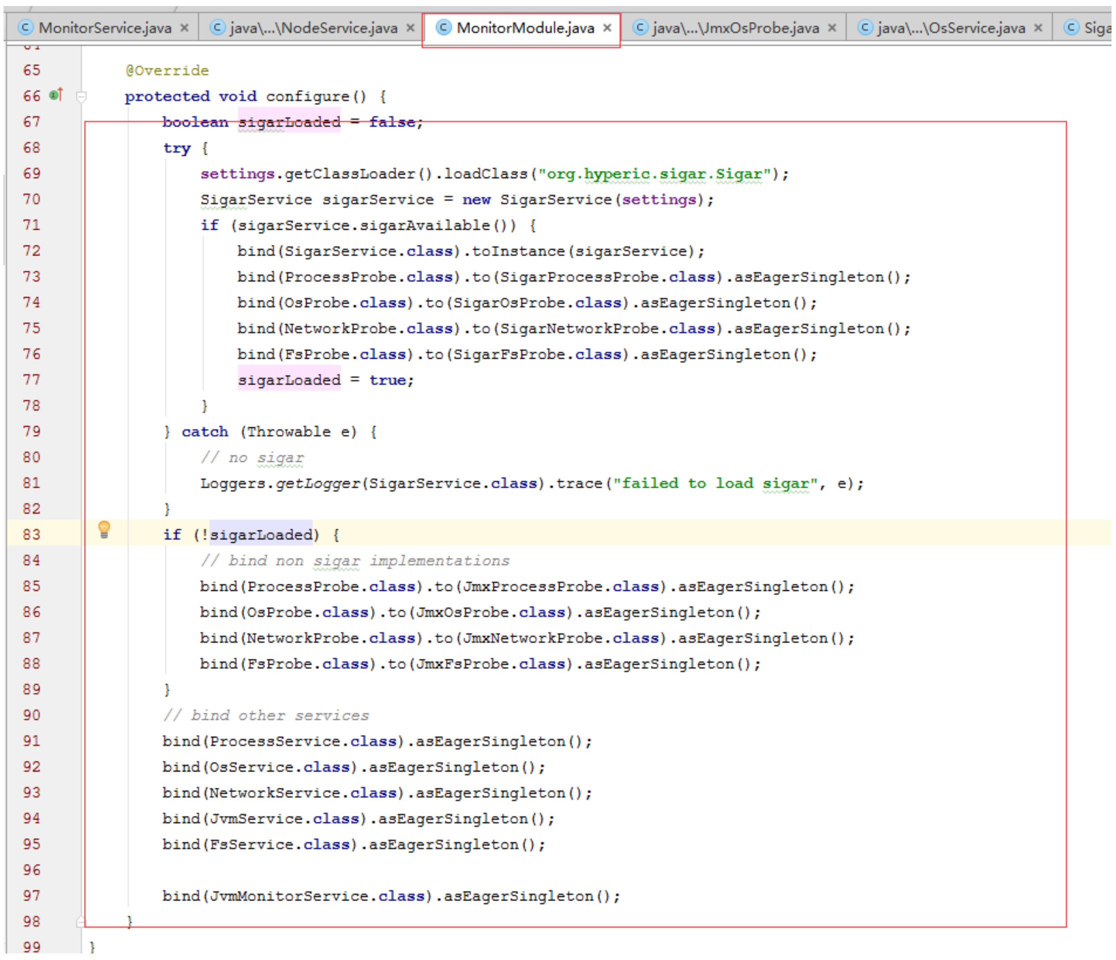
```java
public class MonitorService extends AbstractLifecycleComponent<MonitorService> {

    private final JvmMonitorService jvmMonitorService;

    private final OsService osService;

    private final ProcessService processService;

    private final JvmService jvmService;

    private final NetworkService networkService;

    private final FsService fsService;

    @Inject
    public MonitorService(Settings settings, JvmMonitorService jvmMonitorService,
                          OsService osService, ProcessService processService, JvmService jvmService, NetworkService networkService,
                          FsService fsService) {
        super(settings);
        this.jvmMonitorService = jvmMonitorService;
        this.osService = osService;
```



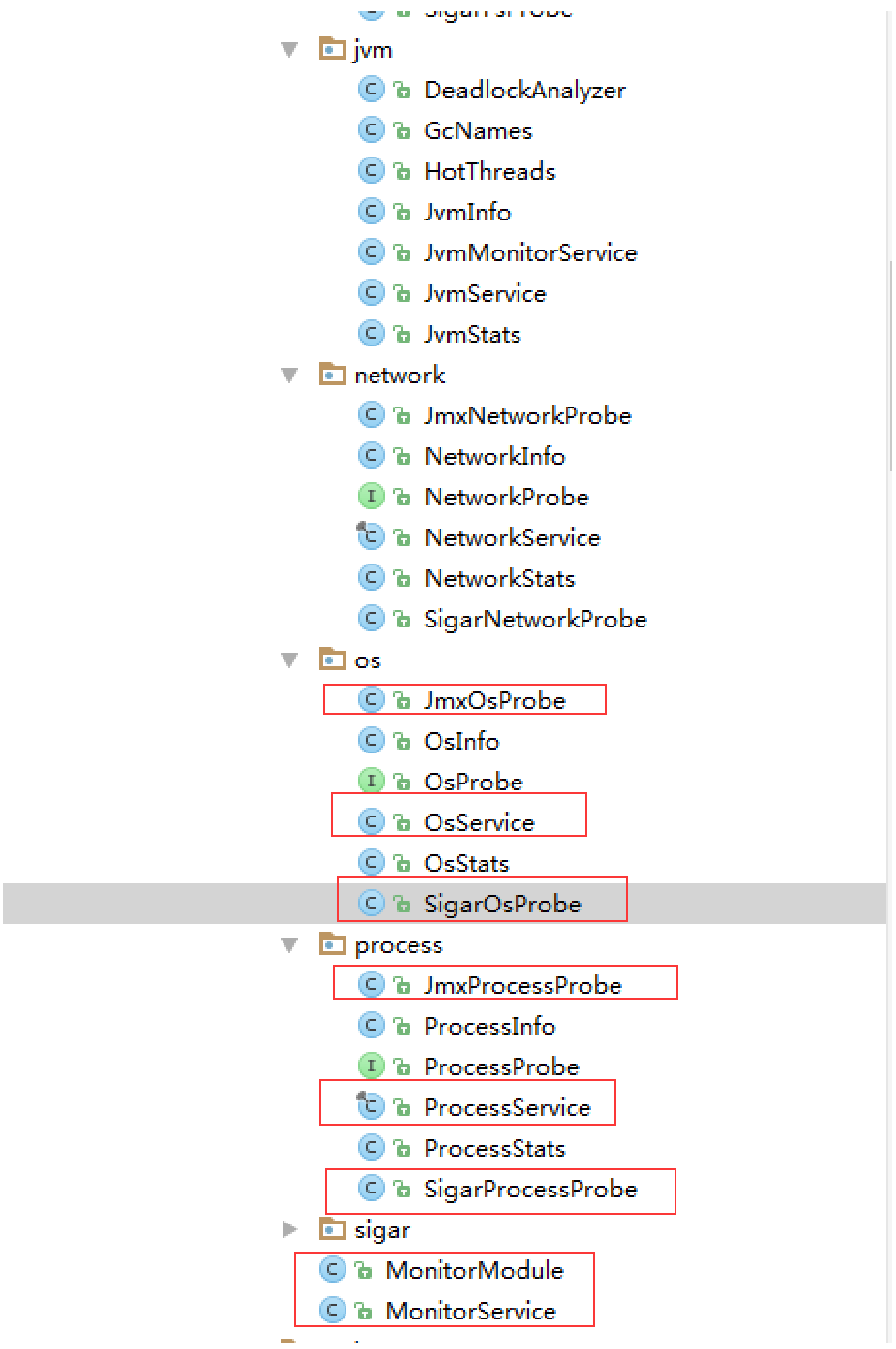
```java
    @Override
    protected void configure() {
        boolean sigarLoaded = false;
        try {
            settings.getClassLoader().loadClass("org.hyperic.sigar.Sigar");
            SigarService sigarService = new SigarService(settings);
            if (sigarService.sigarAvailable()) {
                bind(SigarService.class).toInstance(sigarService);
                bind(ProcessProbe.class).to(SigarProcessProbe.class).asEagerSingleton();
                bind(OsProbe.class).to(SigarOsProbe.class).asEagerSingleton();
                bind(NetworkProbe.class).to(SigarNetworkProbe.class).asEagerSingleton();
                bind(FsProbe.class).to(SigarFsProbe.class).asEagerSingleton();
                sigarLoaded = true;
            }
        } catch (Throwable e) {
            // no sigar
            Loggers.getLogger(SigarService.class).trace("failed to load sigar", e);
        }
        if (!sigarLoaded) {
            // bind non sigar implementations
            bind(ProcessProbe.class).to(JmxProcessProbe.class).asEagerSingleton();
            bind(OsProbe.class).to(JmxOsProbe.class).asEagerSingleton();
            bind(NetworkProbe.class).to(JmxNetworkProbe.class).asEagerSingleton();
            bind(FsProbe.class).to(JmxFsProbe.class).asEagerSingleton();
        }
        // bind other services
        bind(ProcessService.class).asEagerSingleton();
        bind(OsService.class).asEagerSingleton();
        bind(NetworkService.class).asEagerSingleton();
        bind(JvmService.class).asEagerSingleton();
        bind(FsService.class).asEagerSingleton();

        bind(JvmMonitorService.class).asEagerSingleton();
    }
}
```



```java
                network ? monitorService.networkService().info() : null,
                transport ? transportService.info() : null,
                http ? (httpServer == null ? null : httpServer.info()) : null,
                plugin ? (pluginService == null ? null : pluginService.info()) : null
        );
    }

    public NodeStats stats() {
        // for indices stats we want to include previous allocated shards stats as well (it will
        // only be applied to the sensible ones to use, like refresh/merge/flush/indexing stats)
        return new NodeStats(discovery.localNode(), System.currentTimeMillis(),
                indicesService.stats(true),
                monitorService.osService().stats(),
                monitorService.processService().stats(),
                monitorService.jvmService().stats(),
                threadPool.stats(),
                monitorService.networkService().stats(),
                monitorService.fsService().stats(),
                transportService.stats(),
                httpServer == null ? null : httpServer.stats(),
                circuitBreakerService.stats()
        );
    }
```



```java
    @Override
    protected void configure() {
        boolean sigarLoaded = false;
        try {
            settings.getClassLoader().loadClass("org.hyperic.sigar.Sigar");
            SigarService sigarService = new SigarService(settings);
            if (sigarService.sigarAvailable()) {
                bind(SigarService.class).toInstance(sigarService);
                bind(ProcessProbe.class).to(SigarProcessProbe.class).asEagerSingleton();
                bind(OsProbe.class).to(SigarOsProbe.class).asEagerSingleton();
                bind(NetworkProbe.class).to(SigarNetworkProbe.class).asEagerSingleton();
                bind(FsProbe.class).to(SigarFsProbe.class).asEagerSingleton();
                sigarLoaded = true;
            }
        } catch (Throwable e) {
            // no sigar
            Loggers.getLogger(SigarService.class).trace("failed to load sigar", e);
        }
        if (!sigarLoaded) {
            // bind non sigar implementations
            bind(ProcessProbe.class).to(JmxProcessProbe.class).asEagerSingleton();
            bind(OsProbe.class).to(JmxOsProbe.class).asEagerSingleton();
            bind(NetworkProbe.class).to(JmxNetworkProbe.class).asEagerSingleton();
            bind(FsProbe.class).to(JmxFsProbe.class).asEagerSingleton();
        }
        // bind other services
        bind(ProcessService.class).asEagerSingleton();
        bind(OsService.class).asEagerSingleton();
        bind(NetworkService.class).asEagerSingleton();
        bind(JvmService.class).asEagerSingleton();
        bind(FsService.class).asEagerSingleton();

        bind(JvmMonitorService.class).asEagerSingleton();
    }
}
```



```
▼ jvm
    DeadlockAnalyzer
    GcNames
    HotThreads
    JvmInfo
    JvmMonitorService
    JvmService
    JvmStats
▼ network
    JmxNetworkProbe
    NetworkInfo
    NetworkProbe
    NetworkService
    NetworkStats
    SigarNetworkProbe
▼ os
    JmxOsProbe
    OsInfo
    OsProbe
    OsService
    OsStats
    SigarOsProbe
▼ process
    JmxProcessProbe
    ProcessInfo
    ProcessProbe
    ProcessService
    ProcessStats
    SigarProcessProbe
▶ sigar
  MonitorModule
  MonitorService
```