初始化监控服务

```
chain.boot()
        .setNext(new CacheManagerBoot())//1.初始化缓存模块
        .setNext(new ServiceRegistryBoot())//2.启动服务注册与发现模块
        .setNext(new ServiceDiscoveryBoot())//2.启动服务注册与发现模块
        .setNext(new ServerBoot(mPushServer.getConnectionServer(), mPushServer.getConnServerNode(
        .setNext(() -> new ServerBoot(mPushServer.getWebsocketServer(), mPushServer.getWebsocketS
        .setNext(() -> new ServerBoot(mPushServer.getUdpGatewayServer(), mPushServer.getGatewaySe
        .setNext(() -> new ServerBoot(mPushServer.getGatewayServer(), mPushServer.getGatewayServe
        .setNext(new ServerBoot(mPushServer.getAdminServer(), null))//7.启动控制台服务
        .setNext(new RouterCenterBoot(mPushServer))//8.启动路由中心组件
        .setNext(new PushCenterBoot(mPushServer))//9.启动推送中心组件
        .setNext(() -> new HttpProxyBoot(mPushServer), CC.mp.http.proxy_enabled)//10.启动http代理服
        .setNext(new MonitorBoot(mPushServer))//11.启动监控服务
        .end();
```

启动服务

```
public final class MonitorBoot extends BootJob {

    private final MPushServer mPushServer;

    public MonitorBoot(MPushServer mPushServer) {
        this.mPushServer = mPushServer;
    }

    @Override
    protected void start() {
        mPushServer.getMonitor().start();
        startNext();
    }

    @Override
    protected void stop() {
        stopNext();
        mPushServer.getMonitor().stop();
    }
}
```

调用MonitorService->BaseService#start()，然后start()最终调用子类MonitorService#dostar()

MonitorService doStart()

```
    @Override
    public void run() {
        while (isRunning()) {
            MonitorResult result = collector.collect();

            if (printLog) {
                Logs.MONITOR.info(result.toJson());
            }

            if (dumpEnabled) {
                dump();
            }

            try {
                TimeUnit.SECONDS.sleep(dumpPeriod);
            } catch (InterruptedException e) {
                if (isRunning()) stop();
            }
        }
    }

    @Override
    protected void doStart(Listener listener) throws Throwable {
        if (printLog || dumpEnabled) {
            thread = Utils.newThread(ThreadNames.T_MONITOR, this);
            thread.setDaemon(true);
            thread.start();
        }
        listener.onSuccess();
    }
```

如果日志打印或者堆转储启用，则创建守护线程来处理监控信息；